MAX IMS/UTIL

MAX IMS/UTIL Batch V3.4.0

User Reference Manual

MAX SOFTWARE™
Denver, Colorado
MAXimizing your performance

CA, INC.

Contact Information

Corporate Headquarters

CA, Inc.
One CA Plaza
Islandia, NY 11749
USA

www.ca.com

Technical Support

For online technical assistance and a complete list of locations, primary service hours, and telephone numbers, contact Technical Support at http://ca.com/support

Web Site

http://ca.com/support

Disclaimer

Disclaimer of Warranties and Limitation of Liabilities

The staff of MAX Software has taken due care in preparing this manual; however, nothing contained herein modifies or alters in any way the standard terms and conditions of the MAX Software, Inc. standard software evaluation agreement; purchase agreement; lease agreement; or rental agreement by which this software was acquired; nor increases in any way MAX Software's liability to the customer. In no event shall MAX Software be liable for incidental or consequential damages in connection with or arising from the use of this manual or any program contained herein.

Release 3.4.0 (November 2005)

MVS, DB2, IMS, REXX, TSO/E, and ISPF are software products of International Business Machines Corporation.

MAX MVS/UTIL (a complete set of data file manipulation tools with the following 3 components: MAX Data/Util, MAX/PDF and MAX/Batch); MAX IMS/UTIL (a complete set of IMS database manipulation tools with the following 2 components: MAX/IMS Online and MAX/IMS Batch); MAX DB2/UTIL (a complete set of DB2 database manipulation tools); and MAX/REXX (an interface between REXX and VSAM, SAM, PDS and DB2 data) are trademarks of MAX Software, Inc.

TABLE OF CONTENTS

LIST OF FIGURES

REVISIONS

Release 3.4.0: November 2005

Release 3.3.0: March 2004

- Code page selection for data transformation, both input and output.
- Enhanced speed of data transformation.
- Ability to override return codes at the command level.
- Ability to bypass commands based upon return codes.
- Support for external change data.
- Ability to **SORT** files and database segments.
- Ability to output IMS concatenated keys only.
- Ability to output IMS concatenated key along with segment data.
- Ability to delete segments.
- Ability to insert and replace individual segments by concatenated key.
- Database record selection by related root key match.

Release 3.2.0: March 2003

- Added Dynamic PSB generation for accessing a database with MAX IMS/UTIL using only information from the Database Descriptor (DBD) module.
- Added optimized Dynamic PSB definition for BATCH functions to include only those segments in the PSB that are necessary to perform the operation as specified by the select/change criteria resulting in significant performance improvements.
- Added specification of Dynamic PSB segment contents using DYNAMSEG file for BATCH functions resulting in significant performance opportunities.
- Added dynamic PSB generation of Database Descriptor (DBD) defined secondary indexes for accessing a database in a defined alternate processing sequence.
- Export of IMS data mapped by COBOL/PL/I copybooks to XML, comma separated (CSV), tab delimited (TAB), and other user-defined formats. Transformed data may be copied to sequential data sets or Unix System Services files.

Release 3.1.0: June 2002

- Powerful new operands to control the translation or the encoding/decoding of data.
  - **TRANSLATE**
  - **SCRAMBLE**
  - **UNSCRAMBLE**
  - **PIN**
- Additional hex print format.
- Support for Thai character set.
- Default mapping criteria.

Release 2.5.0: July 2001

- Improved format for printing.
- Improved performance for formatted print of records from multiple record format files.
- Improved IMS command performance.

Initial Release 2.4.0: August 2000

- Initial release.
- Added concatenated Copylib support.
- When selected application databases are accessed or changed supports logging to an IMS database.

PREFACE

This book provides a guide and reference about the various functions of MAX IMS/UTIL Batch. Use this book to learn and use the MAX IMS/UTIL Batch product.

It describes:

- An introduction to the product.
- Guideline information on coding statements.
- Command syntax and descriptions.
- Command operand syntax along with a description of the Operands.
- Numerous examples.
- Return code information.
- Cross-reference of Commands to Operands.

Who Should Read This Book

This book is for programmers, database administrators, system programmers, or other technical persons who work with the IMS databases.

Database manipulation includes segment selection, modification and printing. Users are expected to have knowledge of IMS database concepts, MVS JCL, COBOL and/or PL/I.

Notational Conventions

The following notational conventions are used in this manual:

- Uppercase commands and their operand(s) should be entered as shown but need not be in upper case.
- Operand(s) shown in lower case are variables and a value should be substituted for them.
- Operand(s) shown in brackets [ ] are optional, with choices indicated by a vertical bar |. One or none may be chosen; the defaults are underscored.
- Operand(s) shown in braces { } are alternatives; one must be chosen.
- An ellipsis (...) indicates that the parameter shown may be repeated to specify additional items of the same category.

How This Book is Organized

This book contains the following chapters:

Chapter 1: Introduction to MAX IMS/UTIL Batch
describes the need for MAX IMS/UTIL Batch and its overall benefits and uses.

Chapter 2: General Concepts
gives an overview of how the product functions.

Chapter 3: Commands
describes the various commands that may be invoked.

Chapter 4: Command Operands
describes the various command operand(s) which may be specified.

Chapter 5: Examples
shows several samples that invoke various commands and operand(s).

Appendix A: Return Codes
lists the various return codes and their meaning along with where to find additional information about them.

Appendix B: Command Cross-Reference
lists valid Operands for each command.

Appendix C: Copybook Support
lists support details for COBOL and PL/I copybooks.

Appendix D: Changing Installation Defaults
describes how to change the defaults that exist in MAX IMS/UTIL Batch.

Appendix E: Security Considerations
describes the security considerations that exist in MAX IMS/UTIL Batch.

CHAPTER 1: INTRODUCTION TO MAX IMS/UTIL BATCH

MAX IMS/UTIL Batch provides support for the following types of IMS databases and operating environments:

- HISAM, SHISAM, HDAM, HIDAM, HALDB
- FASTPATH DEDB
- Fixed or variable length segment size
- Maximum segment size of 32760
- Uses DL/1 standard commands
- Secondary indexes
- Logical databases and relationships
- Dynamic PSBs
- Operates as a standard IMS Batch Message Processor (BMP) or in DLI Batch mode
- Supports standard existing security packages
- VSAM data sets of the types KSDS, ESDS, RRDS, and VRRDS
- SAM data sets created as the output of a COPY (unload) of a database
- UNIX File System, output only

MAX IMS/UTIL Batch is a command driven utility that performs basic manipulation and printing of an IMS database.

This utility can be invoked as a TSO command or with batch JCL

Commands are specified to perform major processes such as Copy, Print etc.

Command Operands are specified following the command to further control the process such as: record selection, segment selection and segment manipulation, etc.

The following table is a list of tasks that can be performed with MAX IMS/UTIL Batch:

Command	Task	Task Function
ACCUM	Accumulate totals	Accumulate and print values from selected segments.
CALCDAYS	Calculate dates	Calculate the days between two date fields.
COMPARE	Compare	Compare the contents of 2 databases within one PSB or compare the contents of a database with an "unloaded" sequential file.
PRINT FPRINT LIST DUMP	Print	Print selected segments or entire database records from a data set or database.
COPY	Reload	Reload selected segments or database records from a previously unloaded copy of the database into a database.
COPY	Unload	Unload selected segments, selected database records or the entire database to a sequential file. This file will be formatted to allow it to be used as input to load the database.
DELETE	Delete	Delete selected segments in a database.
OPTION	Option	Set return codes and test conditions for commands that follow in the same step.
SORT	Sort	Sort database segments and output files after any file manipulation has occurred.
UPDATE	Update	Update selected segments in a database.

Use of COBOL/PL/I Layout

MAX IMS/UTIL Batch allows for an externally defined COBOL, or PL/I, layout to be used. This layout is used by MAX IMS/UTIL Batch to print an easy-to-read formatted report of data segments. MAX IMS/UTIL Batch will convert all data based upon the field format specified in the file layouts. Refer to "*Appendix C: Copybook Support* on page 117 for more specific information on copybook support.

CHAPTER 2: GENERAL CONCEPTS

IMS Format

MAX IMS/UTIL Batch is designed to allow the user to process data either from an IMS database or process the data in an unloaded format from a sequential file. If the data to be processed resides in a database, it is identified with one of the following operands **INPCB(...)**, **OUTPCB(...)** or **COMPPCB(...)**. By using one of these operands to identify the data, subsequent processing of that data is done in what is referred to in this manual as "IMS-format". Use of the "SEG" option of the **IF/AND/OR** operands allows the user to test for specific segments in the database.

Data copied from an IMS database to a sequential file by MAX IMS/UTIL Batch is said to be in unloaded format. This file contains one record per segment. Each record on the unloaded file is prefixed with control information about the segment. The data on these files is processed using the **INFILE(...)**, **OUTFILE(...)**, and **COMPFILE(...)**, operands. Unloaded files can subsequently be used to reload a database.

MAX IMS/UTIL Batch processes these unloaded files taking the prefix into consideration. The default **FORM** Operand value of "IMS" allows MAX IMS/UTIL Batch to be knowledgeable about the segment and its prefix. Position numbers used in all other Operands reflect the true position within the segment, and use of the **IF** operand with the 'SEG' option is allowed.

Additional **FORM** operand values of **IMSCKEY** and **IMSCSEG** permit unloading database segments along with the value of the segment concatenated key. **IMSCKEY** includes the standard prefix (as in IMS FORM) and only the concatenated key value. **IMSCSEG** includes the IMS prefix (16 bytes), segment concatenated key (255 bytes), and segment data contents. **IMSCSEG** format may be used to update or add any segment in the hierarchy back into the database.

Segments can also be copied from the database to a sequential file without the control prefix. Refer to the **FORM** operand for more specific detail.

IMS Mode

MAX IMS/UTIL Batch provides for processing data at the database record level. Use of the IMS option on commands such as **LIST(LISTIMS)** or **COPY(COPYIMS)** allows you to select entire database records. Commands used in this format are valid only when processing data from the database identified by using **INPCB(...)** for the primary command input. When processing in IMS mode, once a segment is selected, the entire database record, the root with all of its dependent segments, is processed by the primary command.

Command Syntax

```
command [command_operands ...]
```

Parameters

command	Specifies a valid command. (The valid commands are listed in Chapter 3: Commands.) A command is specified before any command operands and may begin in any column. A command initiates processing of an **INFILE** or an **INPCB**.
command_operands	Command operands immediately follow the command. Command operands are processed in the order they are specified. Certain command operands need only be specified once. While other operands such as **IF(..)** may be specified more than once. Command operands are separated by a space character. Each command operand begins with a keyword and its sub-operands are enclosed in parentheses. The sub-operands are separated by commas. Command operands either specify global settings for the command or indicate special processing to be performed on records. Example: **LISTIMS INPCB(PCB01) IF(SEG,EQ,C'BACKORDR')**

Command Continuation

Commands are continued on the next control record by placing a comma following the last command operand on a control record. An entire command operand (that is the keyword and its parenthesized sub-operands) must fit on one control record. A continued control record can begin in any column. However, using indentation can make the entire command more readable.

Example:

```
0 ---- + ----1---- + ---- 2 ---- + ---- 3 ---- + ---- 4 ---- + ---- 5 ---- + ---- 6 ---- + ---- 7 ---- + ---- 8
          COPYIMS,
                    IF(SEG,EQ,C'segname'),
                    REPLACE(50,C'P')
```

Comments

An '*' (asterisk) character in column 1 designates a comment record. Also, comments may be specified following the comma placed after a command operand.

Example:

Unload all segments replacing column 50 with a 'P' for a segment named 'ROOTSEG'.

```
COPYALL INPCB(PCB01),          Unload a PCB01
OUTFILE(SYSUT2),               to SYSUT2
IF(SEG,EQ,C'ROOTSEG'),         IF segment name = ROOTSEG
REPLACE(50,C'P');              Replace column 50 with "P"
```

Command Termination

A complete command (with its associated command operands) is determined when a comma is not detected following the last command operand on a control record or when a semi-colon is detected. By using a semi-colon, a comment may be placed on the last control record for a specific command.

Example:

```
COPY INPCB(PCB01) OUTFILE(BACKUP);  /* Unload database */
```

Multiple Commands

Multiple commands may be processed in a single execution of MAX IMS/UTIL Batch. The input PCB or input file, as identified by the **INPCB(..)** or **INFILE(..)** operand is closed and repositioned to the beginning between command execution. It is possible to not close and reposition the input by using the **FORM (MULTI)** operand. Refer to the **FORM** operand for details.

A new command begins when a previous command ends. The end of a command is determined when either there is no continuation specified or a semi-colon has been specified.

Example:

```
COPYIMS INPCB(PCB01),                 /*Copy entire records                    */
        OUTFILE(SYSUT2),              /*To SYSUT2 file                         */
        IF(SEG,EQ,C'BACKORDR');       /*If they contain a backorder segment    */
LIST INFILE(SYSUT2),                  /*List the unloaded file                 */
        IN(100);                      /*The first 100 records                  */
```

Allocations

The following DDNAMES are used by MAX IMS/UTIL Batch.

Ddnames	Requirement	Description
SYSIN	May be Optional	Control records input file. If all defaults are being used, code SYSIN as a DD DUMMY statement.
SYSLIST	Optional	Record printing output file. When not allocated this data will be written to SYSPRINT.
SYSTSPRT	Required	Messages.
SYSPRINT	Required	Messages.
SYSTOTAL	Optional	Accumulated totals. When not allocated this data will be written to SYSPRINT.
IMS	Required	Concatenation of libraries containing DBD and PSB load modules. Specifies the names of the partitioned data sets (DSORG=PO) containing standard DBD and PSB load modules defined for each IMS database to be accessed. These load modules are produced from the standard output of the DBDGEN and PSBGEN processes. When Dynamic PSBs are used, only the DBD library containing the DBD modules is required.

Ddnames	Requirement	Description
COPYLIBS	Optional	Concatenation of libraries containing Copybook Members. When a segment layout definition (MAPDD operand) specifies a 'copybook' name without a data set name, the member is retrieved from the libraries allocated to this DD name. **COMPARE** and **FPRINT** can use the **MAPDD** operand to format using a segment layout definition. Default mapping criteria also retrieve copybook members from this library.
DYNAMSEG	Optional	Only applies when Dynamic PSBs are used. Control record input file containing segment names to be included in the dynamically generated PSB. Each segment name is included on a separate record in position 1-8. Requested segments and all higher level segments along the concatenated key will automatically be included in the PSB definition. All functions will operate on only the chosen segments.
SYSUT1 **SYSLIN** **SYSLIB** **SYSLMOD**	Reserved	With Dynamic PSBs, these ddnames are allocated to assemble and link the generated PSB and must not be used in any command operands (e.g., INFILE, OUTFILE, etc.).

The following defaults are used as input/output to MAX IMS/UTIL Batch when no specific input (using **INPCB(..)** or **INFILE(..)** operand) or output for a copy command (using **OUTPCB(..)** or **OUTFILE(..)** operand) has been specified.

Default	Description
INPCB(#1)	If no input is specified, the input will default to be the first PCB in the PSB.
OUTPCB(#1)	If the command is a form of **COPY** and no output is specified, the output will default to the first PCB in the PSB
COMPPCB(#1)	If the command is a form of **COMPARE** and no compare data is specified, the compare data will default to the first PCB in the PSB.

Accessing IMS Databases

Generally, accessing IMS databases with MAX IMS/UTIL Batch requires entry of an IMS subsystem id, PSB name, PCB/DBD name, and IMS run type to identify the IMS database to be processed and whether to run in BMP or DLI batch mode.

An existing Static PSB name may be used or a Dynamic PSB can be generated from an entered DBD name by specifying DYNAM or DYNAMSEG in the PSB name field. Using Dynamic PSBs can offer significant performance benefits, particularly in batch functions, where only necessary segments are included in the generated PSB.

Dynamic PSBs require parameters to be setup in the MAXIOPTS installation options module before they can be used. See the MAX Product Installation Guide for information about setting the installation options for Dynamic PSB.

Using an Existing Static PSB

IMS subsystem id, PSB name, PCB/DBD name, and IMS run type are specified to identify an IMS database to be processed and whether to run in BMP or DLI batch mode.

PSB name must correspond to a valid IMS PSB in the connected IMS Subsystem. Additionally, the load module output from the PSBGEN and DBDGEN must be available in the PSB and DBD libraries allocated with the IMS ddname. A valid database PCB name, DBD name, or relative DB PCB number (#n) from the PSB must be specified in the **INPCB()**, **OUTPCB()**, or **COMPPCB()** operands to identify the actual database to be processed.

BMP Execution Requirements with Static PSBs

MAX IMS/UTIL Batch may run as a standard IMS Batch Message Processor (BMP) program when using existing static PSBs.

The parameter values that will be specified to the MAXRRC00 program and passed to the IMS DFSRRC00 region controller program are defined here. The IMS positional parameters are described in detail in the IMS Install Manual, Volume 2. See the IMSBATCH procedure parameter discussion about the following values:

```
PARM=(BMP,#MBR,#PSB,#IN,#OUT,#OPT#SPIE#TEST#DIRCA,#PRLD,#STIMER,
#CKPTID,#PARDLI,#CPUTIME,#NBA,#OBA,#IMSID,#AGN,#SSM,#PREINIT,
#ALTID,#APARM,#LOCKMAX)
```

Several of these parameters must be specified as follows:

Parameter	Description
#MBR	Program to be executed. This must be MAXIBAT.
#PSB	PSB name to be processed. Any PSB defined to be processed as a BMP may be specified.
#IMSID	IMS subsystem name to which to connect.
#AGN	Application group name specifies a one-to-eight character group name for inter-regional security to which you are authorized.

All other parameters may be set to your installation requirements.

Sample JCL

This sample will unload selected database records from PCB01 in PSB MAXIBRED to SYSUT2 in BMP mode.

```
//SAMPJOB JOB 0,'IMS UNLOAD',
//            CLASS=A,MSGCLASS=D
//* * * * * * * * ** * * * * * * * * * * * * * * * * * * * * * * * *
//* DOC: UNLOAD SELECTED DATABASE RECORDS - BMP Mode
//* * * * * * * * ** * * * * * * * * * * * * * * * * * * * * * * * *
//STEP1   EXEC PGM=MAXRRC00,REGION=4M,
//          PARM=(BMP,MAXIBAT,MAXIBRED,,,C00000,,,,1,,15,15,IVP3,
//          IVP,,,,,0)
//STEPLIB   DD DISP=SHR,DSN=MXS.MXRXV310.LOADLIB
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//          DD DSN=IMS.PSBLIB,DISP=SHR
//SYSTSPRT  DD SYSOUT=*              SYSPRINT
//SYSLIST   DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSUT2    DD DSN=IMS.PCB.UNLOAD,UNIT=SYSDA,
//  SPACE=(CYL,(10,2)),DCB=(RECFM=VB,LRECL=200,BLKSIZE=8004)
//SYSIN DD  *
 COPYIMS INPCB(PCB01) OUTFILE(SYSUT2),
   IF(SEG,EQ,C'STOKSTAT') AND(4,EQ,C'AA16')
```

Figure 1: JCL COPY selected database records (COPYIMS) in BMP

DLI BATCH Execution Requirements with Static PSBs

MAX IMS/UTIL Batch may run offline in DL/1 Batch mode when using existing static PSBs.

The parameter values that will be specified to the MAXRRC00 program and passed to the IMS DFSRRC00 region controller program are defined here. The IMS positional parameters are described in detail in the IMS Install Manual, Volume 2. See the DLIBATCH procedure parameter discussion about the following values:

```
PARM =(DLI,#MBR,#PSB,#BUF,#SPIE#TEST#EXCPVR#RST,#PRLD,#SRCH,
#CKPTID,#MON,#LOGA,#FMTO,#IMSID,#SWAP,#DBRC,#IRLM,#IRLMNM,#BKO,
#IOB,#SSM,#APARM,#LOCKMAX,#GSGNAME,#TMINAME)
```

#MBR, #PSB, and #IMSID are specified the same as in the BMP parameter description (see the Parameter table on page 9). All other parameters may be set to your installation requirements.

Sample JCL

This sample will unload selected database records from PCBO1 in PSB MAXIBRED to SYSUT2 in DLI BATCH mode.

```
//SAMPJOB JOB 0,'IMS UNLOAD',
//             CLASS=A,MSGCLASS=D
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//* DOC: UNLOAD SELECTED DATABASE RECORDS - DLI BATCH MODE
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//STEP1   EXEC PGM=MAXRRC00,REGION=4M,
//          PARM=(DLI,MAXIBAT,MAXIBRED,7,0000,,0,,N,,T,IVP3,
//          Y,N,N,,N,,,,0,,)
//STEPLIB   DD DISP=SHR,DSN=MXS.MXRXV310.LOADLIB
//IMS       DD DISP=SHR,DSN=IMS.PSBLIB
//          DD DISP=SHR,DSN=IMS.DBDLIB
//DFSVSAMP  DD DISP=SHR,DSN=IMS.PROCLIB(DFSVSMDB)
//IEFRDER   DD DUMMY
//IEFRDER2  DD DUMMY
//SYSTSPRT  DD SYSOUT=*
//SYSLIST   DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSUT2    DD DSN=IMS.PCB.UNLOAD,UNIT=SYSDA,
//             SPACE=(CYL,(10,2)),DCB=(RECFM=VB,LRECL=200,BLKSIZE=8004)
//SYSIN DD *
   COPYIMS INPCB(PCB01) OUTFILE(SYSUT2),
     IF (SEG,EQ,C'STOKSTAT') AND(4,EQ,C'AA16')
```

Figure 2: JCL COPY selected database Records (copyims) in DLI BATCH MODE

Other DD statements may be required to support specific DLI parameter options that you have chosen (i.e. logging data sets).

RESLIB data sets should also be specified if not previously placed in the LINKLIST or LPA.

Using a Dynamic PSB

When DYNAM or DYNAMSEG are specified as the PSB name, a PSB is dynamically built and generated from the segment names contained in the entered DBD name. A single PSB is generated and used for duration of the entire MAX IMS/UTIL Batch run. The segments included and the PROCOPT generated in the dynamic PSB are a result of analyzing ALL commands and operands for the entire MAX IMS/UTIL Batch run.

In addition to a PSB name of DYNAM or DYNAMSEG, the IMS subsystem id, DBD name, Index name (optional), and IMS run type must be entered to identify an IMS database to be processed and whether to run in BMP or DLI batch mode. An IMS database may also be accessed through a valid secondary index by appending the PROCSEQ index name to the DBD name separated by a '.'

DBD name and index name must correspond to a valid IMS DBD in the connected IMS Subsystem. Additionally, the load module output from the DBDGEN must be available in the DBD library allocated to the IMS ddname.

Read only commands generate a PROCOPT=G, while update commands generate a PROCOPT=A (or R if an alternate index is being used).

DYNAM will include only those segments contained in the DBD that are required by the specified command and segment selection/change criteria. All higher level segments along the concatenated key will automatically be included in the PSB definition. If no segment selection/change criteria has been specified or the command ends in 'IMS' or 'ALL', all DBD segments are included in the generated PSB.

Message RXPS019I will identify the PROCOPT and PSB segment names included as follows.

```
* RXPS019I DYNAMIC PSB SEGMENTS DETERMINED FROM BATCH COMMAND ANALYSIS
*            PROCOPT=G   PSB SEGMENTS: PARTROOT
```

DYNAMSEG will include only those segments from the DBD that have been specified in the DYNAMSEG ddname control record file. Each segment name to be included is specified in position 1-8 of a separate DYNAMSEG control record. All higher level segments along the concatenated key will automatically be included in the PSB definition. All commands and operands will operate on only the DYNAMSEG requested segments. Segment names contained in selection/change criteria must also be in the generated PSB or the run will fail.

Message RXPS018I will identify the PROCOPT and PSB segment names included as follows:

```
* RXPS018I DYNAMIC PSB SEGMENTS DETERMINED FROM DYNAMSEG FILE
             PROCOPT=G   PSB SEGMENTS: PARTROOT STANINFO
```

BMP Execution Requirements with Dynamic PSBs

MAX IMS/UTIL Batch may run as a standard IMS Batch Message Processor (BMP) program when using dynamic PSBs.

The parameter values that will be specified to the MAXRRC00 program and passed on to the IMS DFSRRC00 region controller program are defined here. The IMS positional parameters are described in detail in the IMS Install Manual, Volume 2. See the IMSBATCH procedure parameter discussion about the following values:

```
PARM=(BMP,#MBR,#DYNTYP.#DBD.#INDEX,#IN,#OUT,#OPT#SPIE#TEST#DIRCA,
#PRLD,#STIMER,#CKPTID,#PARDLI,#CPUTIME,#NBA,#OBA,#IMSID,#AGN,#SSM,
#PREINIT,#ALTID,#APARM,#LOCKMAX)
```

Several of these parameters must be specified as follows:

Parameter	Description
#MBR	Program to be executed. This must be MAXIBAT.
#DYNTYP	Dynamic PSB type must be DYNAM or DYNAMSEG indicating a dynamic PSB is to be generated from #DBD
#DBD	DBD name to be processed appended to the word DYNAM or DYNAMSEG separated by a '.'.
#INDEX	Optional PROCSEQ index name appended to the DBD name separated by a '.'.
#IMSID	IMS subsystem name to which to connect.
#AGN	Application group name specifies a one-to-eight character group name for inter-regional security to which you are authorized.

All other parameters may be set to your installation requirements.

Sample JCL

This sample will copy selected STOKSTAT segments from DBD MAXDBD1 to SYSUT2 in BMP mode using dynamic PSBs. The generated dynamic PSB will only contain segment names for STOKSTAT and its hierarchical parent PARTROOT.

```
//SAMPJOB JOB  0,'IMS UNLOAD',
//             CLASS=A,MSGCLASS=D
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//* DOC: UNLOAD SELECTED DATABASE SEGMENTS - BMP MODE
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//STEP1   EXEC PGM=MAXRRC00,REGION=4M,
// PARM=('BMP,MAXIBAT,DYNAM.MAXDBD1',
// ,,C00000,,,,1,,15,15,IVP1,IVP,,,,,0)
//STEPLIB  DD  DSN=MXS.MXRXV320.LOADLIB,DISP=SHR
//         DD  DSN=IMS710.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS710.SDFSRESL,DISP=SHR   NOTE: Required for ACBGEN
//IMS      DD  DSN=IMS.DBDLIB,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSLIST  DD  SYSOUT=*
//SYSUT2   DD  DSN=IMS.DB.UNLOAD,UNIT=SYSDA,
//     SPACE=(CYL,(10,2)),DCB=RECFM=VB,LRECL=32756,BLKSIZE=32760)
//SYSIN    DD  *
 COPY INPCB(MAXDBD1) OUTFILE(SYSUT2),
   IF(SEG,EQ,C'STOKSTAT') AND(4,EQ,C'AA16')
```

Figure 3: JCL COPY selected database segments in BMP using dynamic PSB

DLI BATCH Execution Requirements with Dynamic PSBs

MAX IMS/UTIL Batch may run offline in DL/I Batch mode when using dynamic PSBs.

The parameter values that will be specified to the MAXRRC00 program and passed on to the IMS DFSRRC00 region controller program are defined here. The IMS positional parameters are described in detail in the IMS Install Manual, Volume 2. See the DLIBATCH procedure parameter discussion about the following values:

```
PARM=(DLI,#MBR,#DYNTYP.#DBD.#INDEX,#BUF,#SPIE#TEST#EXCPVR#RST,
#PRLD,#SRCH,#CKPTID,#MON,#LOGA,#FMTO,#IMSID,#SWAP,#DBRC,IRLM,
#IRLMNM,#BKO,#IOB,#SSM,#APARM,#LOCKMAX,#GSGNAME,#TMINAME)
```

#MBR, #DYNTYP, #DBD, #INDEX, and #IMSID are specified the same as in the BMP parameter description (see Parameter table on page 13). All other parameters may be set to your installation requirements.

Sample JCL

This sample will unload database records from DBD MAXDBD1 to SYSUT2 in DLI BATCH mode using dynamic PSBs. The DYNAMSEG file explicitly defines the segment names to be included in the generated dynamic PSB. As a result, the database record to be unloaded is a subset of the DBD defined database record.

```
//SAMPJOB JOB  0,'IMS UNLOAD',
//             CLASS=A,MSGCLASS=D
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//* DOC: UNLOAD SELECTED DATABASE RECORDS - DLI BATCH MODE
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//STEP1   EXEC PGM=MAXRRC00,REGION=4M,
// PARM=('DLI,MAXIBAT,DYNAMSEG.MAXDBD1',
// 7,0000,,0,,N,,T,IVP1,Y,N,N,,N,,,,0,,)
//STEPLIB  DD  DSN=MXS.MXRXV320.LOADLIB,DISP=SHR
//         DD  DSN=IMS710.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS710.SDFSRESL,DISP=SHR
//IMS      DD  DSN=IMS.DBDLIB,DISP=SHR
//DFSVSAMP DD  DSN=IMS710.PROCLIB(DFSVSMDB),DISP=SHR
//IEFRDER  DD  DUMMY
//IEFRDER2 DD  DUMMY
//DI21PART DD  DSN=IMS710.DI21PART,DISP=SHR      DB file allocation
//DI21PARO DD  DSN=IMS710.DI21PARO,DISP=SHR      DB file allocation
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSLIST  DD  SYSOUT=*
//SYSUT2   DD  DSN=IMS.DB.UNLOAD,UNIT=SYSDA,
//     SPACE=(CYL,(10,2)),DCB=RECFM=VB,LRECL=32756,BLKSIZE=32760)
//DYNAMSEG DD  *
PARTROOT
STOKSTAT
STANINFO
//SYSIN    DD  *
 COPYIMS INPCB(MAXDBD1) OUTFILE(SYSUT2),
   IF(SEG,EQ,C'STOKSTAT') AND(4,EQ,C'AA16')
```

Figure 4: JCL COPY selected database records in DLI using dynamic PSB

Other DD statements may be required to support specific DLI parameter options that you have chosen (i.e. logging data sets)

MAX IMS/UTIL DATABASE LOGGING

MAX IMS/UTIL supports the automatic monitoring of selected databases whenever they are accessed or changed using MAX IMS/UTIL. Logging information is recorded in an IMS HDAM database whenever the MAXLOG logging database PCB is present in the PSB specified for use with MAX IMS/UTIL. A PCB name of MAXLOG identifies the logging database.

For more information concerning the MAXLOG database logging feature, see member MAXLREAD in the MAX IMS/UTIL JCL library.

CHAPTER 3: COMMANDS

Summary

This chapter discusses the following lists of commands used in MAX IMS/UTIL Batch program. A command initiates processing of an **INPCB** or **INFILE**.

Operands may be specified for each command and control the processing of segments within a database by the command.

Accumulation Commands

Command	Description
ACCUM	Accumulate and print values from selected segments.
ACCUMALL	Accumulate and print values from all segments.
ACCUMIMS	Accumulate and print values from selected database records.

ACCUM Command

Accumulate totals for a field.

Command Format

```
{ACCUM|ACCUMALL|ACCUMIMS} [operands ...]
```

ACCUM is used to add all values from a numeric type field from the selected segments and print a report showing the accumulated totals. Normally, one or more **ACCUM(..)** operands are supplied which identify where in the segment the numeric field is located. When the **ACCUM(..)** operand is not supplied, a report is produced showing the total of number of segments read and selected from the database or file.

The report is written to file name SYSTOTAL unless the SYSTOTAL file is not allocated in which case the report is written to SYSPRINT.

Sample **ACCUM** Report

```
ACCUM  INPCB(PCB01)  IF(SEG,EQ,C'STOKSTAT')  ACCUM(34,4,P,'Totals for column 34')
```

The above command produces the following output:

```
TOTAL FOR COLUMN 34  = 12347654
```

Examples of ACCUM

Example 1:

```
ACCUM   INPCB(PCB01)
```

Prints a report showing the total number of segments in the input database PCB01.

Example 2:

```
ACCUMIMS   INPCB(PCB01)
```

Prints a report showing the number of database records in the database.

Example 3:

```
ACCUM   INPCB(PCB01),
            IF(SEG,EQ,C'BACKORDR'),
            ACCUM(25,5,P,'TOTAL COL25, BACKORDER SEGMENTS')
```

Prints a report showing the total value in the packed field at position 25 for segments named 'BACKORDR'.

Calculate the Number of Days Between Two Dates

Command	Description
CALCDAYS	Calculate the number of days between two dates.

CALCDAYS Command

Calculate number of days to use in date aging

Command Format

{CALCDAYS} [operands ...]

CALCDAYS is used to calculate the number of days between two dates. The result may then be used in subsequent **CALCDATE** operands. This command can be used to simplify the aging of date fields to a specific date by eliminating the need to know the exact number of days between the two dates.

Examples of CALCDAYS

Example 1:

```
CALCDAYS   GDATES(2000/04/23,2003/04/23)
COPY  INPCB(PCB01)  OUTFILE(SYSUT2)  IF(SEG,EQ,C'ACCNTDTE'),
                CALCDATE(10,G1ZSECE,*)
```

Advances all dates in a given segment of the database the number of days between the two dates specified. The '*' character in the **CALCDATE** operand of the **COPY** command indicates that the number of days calculated in the **CALCDAYS** command should be used in place of hard coding the number of days difference.

Example 2:

```
CALCDAYS  GDATES(2002/04/23,2000/04/23)
```

In this example, during the **COPY** process, all dates will be advanced the number of days calculated by the **CALCDAYS** command. To cause the dates to be aged in a negative direction, specify the current date first, followed by the prior date, forcing a negative number of days to be calculated.

This would result in a negative number that would be subtracted from any dates during the aging process.

Compare Commands

Command	Description
COMPARE	Compare selected segments between two databases, or a database and a file.
COMPAREALL	Compare all segments between two databases, or a database and a file.

COMPARE Command

COMPARE compares the differences of two inputs. The inputs can be in either database **INPCB(..)**, **COMPPCB(..)** or unloaded format **INFILE(..)** or **COMPFILE(..)**. The **INPCB(..)** or **INFILE(..)** command operand identifies the source of the original data. The **COMPPCB(..)** or **COMPFILE(..)** command operand identifies the source of the new (compare to) data.

Command Format

{COMPARE|COMPAREALL} [operands ...]

A compare report is written to a file name SYSLIST, unless the SYSLIST file is not allocated, in which case the report is written to SYSPRINT.

When **COMPAREALL** is used, a segment that is NOT selected by an **IF** operand is compared in its entirety. See **COMPARE** example in Chapter 5.

```
MAX PRINT SERVICES                              DATE=22 Dec 1999 TIME=08:59:03                          PAGE 1

 SEG MISMATCHES POS=36    STOCK STATUS        INPCB(#1)      SEG#=5 LEN=176              COMPFILE(SYSUT2) REC#=5 LEN=176
 POSS  *----------FIELD NAME---------* FORMAT *---+----1----+----2----+----3----+----4-*
*---+----1----+----2----+----3----+----4-*
 00001 STOKSTAT-FORMAT
 00001 W                              C    2 00
 00003 STOKSTATUS-KEY
 00003 X                              C    8  AK2877F
 00021 U-P                            Z  6.3 000100.000                                  X'E7E7E7E7E7E7F0F0F0'
 00035 UOM                            C    4 EACH
 00051 ATTRITION
 00051 COAP                           Z    3 270
 00054 PLANNED                        Z    3 000
 00057 COAD                           C    1
 00058 /FILLER-5/                     C   32                 360    0000000{
 00090 REQUIREMENTS
 00090 CURRNT                         Z  7.1 +0000088.0
 00098 UNPLANNED                      Z  7.1 +0000000.0
 00106 ON-ORDER                       Z  7.1 +0000000.0
 00114 TOTAL-STOCK                    Z  7.1 0000088.0
 00122 DISBURSEMENTS
 00130 UNPLAN                         Z  7.1 0000000.0
 00138 SPARES                         Z  7.1 +0000000.0
 00146 DIVERS                         Z  7.1 +0000000.0
 00154 /FILLER-7/                     C    7 0 Z360N
```

Figure 5: Formatted Compare Report Output panel

Examples of COMPARE

Example 1:

```
COMPARE  INPCB(#1)  COMPFILE (SYSUT2)
```

Compare the original database (**INPCB**) to an unloaded copy of the database (**COMPFILE**). Print a report to SYSLIST showing the differences.

Example 2:

```
COMPARE  INFILE(PCB01)  COMPFILE(SYSUT2)  MAPDD(IMSMAP)
```

Compare the original database (**INPCB**) to an unloaded copy of the database (**COMPFILE**). Print a report to SYSLIST showing the differences. The output report will be in formatted mode using the specified mapping criteria to map the data segments.

Example 3:

```
COMPARE  INFILE(PCB01)  COMPFILE(SYSUT2),
             IF(SEG,EQ,C'PARTROOT')
```

Compare the original database (**INPCB**) to the unloaded copy of the database (**COMPFILE**). Certain segments are selected with Boolean logic to be included in the compare (non-selected segments will not be compared). Print a report to SYSLIST showing the differences. In this example compare only the root segments named '`PARTROOT`'.

Example 4:

```
COMPAREALL INPCB(PCB01) COMPPCB(PCB02),
             IF(SEG,EQ,C'STOKSTAT') AND(51,EQ,C'270') COMPARE(1,50,1)
             COMPARE(54,0,54)
```

Compare the original database (**INPCB**) to the new database (**COMPPCB**).

Note: When both inputs to the compare are databases, they must be within the same PSB.

If the segment name is '`STOKSTAT`', compare a subset of the segment. If the `STOKSTAT` segment does not contain the value '`270`' in position 51, compare the entire segment.

Note: The segments which are not selected by the **IF** operand will be compared in their entirety.

Example 5:

```
COMPAREALL INPCB(PCB01) COMPFILE(SYSUT2),
             IF(SEG,EQ,C'STOKSTAT') SKIP(0)
```

Compare the original database (**INPCB**) to the unloaded file (**COMPFILE**). Bypass the compare of any segments named '`STOKSTAT`'. The output will be provided on SYSLIST.

Copying and Sort Commands

Command	Description
COPY	Copy selected segments.
COPYALL	Copy all segments.
COPYIMS	Copy selected database records.
SKIP	Skip selected segments while copying.
SORT	Copy and sort selected segments.
SORTALL	Copy and sort all segments.

COPY Command

Load Or Unload A Database. See also SKIP Command.

Command Format

{COPY|COPYALL|COPYIMS} [operands ...]

Use this command to unload a database **INPCB(...)** to an output sequential file **OUTFILE(...)**. Use this command to load a database from an unloaded copy **INFILE(...)** to the database **OUTPCB(...)**. When segments are unloaded using the **COPY** command, they are prefixed with control information which is then used to reload the database.

When using the "IMS mode", **COPYIMS**, selectively unload complete database records based upon selection criteria. Use this mode to create a test database containing specific test data.

Segments can also be copied to an output file without the control prefix. Use the **FORM(NOIMS)** operand to create a file with only the segment data. Segments copied using this option cannot be reloaded into the database using MAX IMS/UTIL Batch.

When unloading segments from a database to a sequential file, it is recommended that you specify the DCB information for the outfile in the JCL. For LRECL, choose the largest segment length of those being unloaded. When using IMS format, it is necessary to add 16 to this length to account for the control information on the unloaded segment. If different size segments are unloaded, the record format must be variable.

Data can be reloaded into a database using the **COPY** command. Input specified using the **INFILE(..)** operand must be a file in IMS format, the output of a **COPY** command using MAX IMS/UTIL Batch.

When copying to a database PCB, complete database records consisting of a root and dependent segments are inserted into the database. The first segment to be copied must be a root segment to establish parentage. Segments containing duplicate keys are deleted and then the new segment inserted. Dependent segments of any deleted duplicate segments are also deleted before the insert. Only segments to which the PSB is sensitive can be loaded.

The PSB PROCOPT must support insert and delete of segments. A PSB PROCOPT of L or LS is not valid unless the database is empty and initialized by VSAM. When loading segments, a SYNCPOINT is issued every 20 database records.

Data can also be copied from one database **INPCB(...)** directly into a second database **OUTPCB(...)**. W

Note: When both the input and output of the copy command are a database, they must be within the same PSB.

Examples of COPY

Example 1:

```
COPY  INPCB(PCB01)  OUTFILE(SYSUT2)
```

Unload all segments from database PCB01 to SYSUT2.

Example 2:

```
COPYIMS  INPCB(PCB01) OUTFILE(SYSUT2),
              IF(SEG,EQ,C'BACKORDR')
```

Unload selected database records (IMS mode) from PCB01 to SYSUT2. The database records selected must contain a backorder segment, 'BACKORDR'.

Example 3:

```
COPYALL  INPCB(PCB01)  OUTFILE(SYSUT2),
              IF(SEG,EQ,C'BACKORDR')  AND(69,EQ,C'1000')   REPL(69,C'4000')
```

Unload all segments from PCB01 to SYSUT2. If the segment named 'BACKORDR' has a value of '1000' in column 69, change this value to '4000' with a **REPLACE** operand.

Example 4:

```
COPY   INPCB(PCB02)   OUTFILE(SYSSUT2)   FORM(NOIMS),
              IF(SEG,EQ,C'STOKINFO')
```

Copy all segments entitled 'STOKINFO' to a sequential file for the purpose of analyzing specific data within the segments. The segments cannot be reloaded to the database.

SKIP Command

Unload a database, skipping selected segments. See also "*COPY Command*" on page 25.

Command Format

```
SKIP [operands]
```

SKIP unloads segments from an input database file to one or more output files. However, any segments meeting the selection criteria of the **IF/AND/OR** operands are skipped. Therefore, the selected segments are skipped when unloading the database or file.

Segments can also be copied using the SKIP command to an output file without the control prefix (format IMS). To copy segments without this information use the **FORM(NOIMS)** operand. Segments copied using this operand cannot be reloaded to a database using MAX IMS/UTIL Batch.

Examples of SKIP

Example 1:

```
SKIP  INPCB(PCB01) OUTFILE(SYSUT2),
            IF(SEG,EQ,C'STOKSTAT')  AND(4,EQ,C'AK2877F')
```

Unload all segments named '`STOKSTAT`' from PCB01 to SYSUT2, unless the value of '`AK2877F`' begins in column 4 of the segment.

Example 2:

```
SKIP  INPCB(PCB01) OUTFILE(SYSUT2),
            IF(SEG,EQ,C'BACKORDR')  AND(20,LT,C'1990')
```

Unload all segments from the input database PCB01 to SYSUT2. For all `BACKORDR` segments, do not unload those segments prior to `1990`.

SORT Command

Copies a file or database, processing records or segments as indicated by the operands and then sorts the output into the specified sequence.

Note: The `OUTFILE(filename)` must reference a non-VIO data set name.

Command Format

```
{SORT|SORTALL} [operands ...]
```

Examples of SORT

Example 1:

```
SORT INPCB(PODB) OUTFILE (SYSUT2),
        IF(SEG,EQ,C'ORDER'),
        REPL(1,C'ABC') CHNG(22,EQ,C'S',CC'J'),
        SORTPARM(SORT FIELDS(7,6,CH,A))
```

Copies all the input PODB database ORDER segments replacing columns `1` to `3` with the character '`ABC`', changing any '`S`' in column `22` to a '`J`' and then sort the file into ascending sequence based on the data in columns `7` through `12`. The output of the **SORT** will be in the file referenced by `OUTFILE`.

Example 2:

```
SORTALL INPCB(INVDB) OUTFILE(SYSUT2),
        IF(SEG,EQ,C'ITEM'),
        ANDIF(1,10,C'ABC'),
        REPL(30,C'DEF')
        SORTPARM(SORT FIELDS(54,3,CH,A))
```

Copies all the input INVDB database segments changing only those that are selected by the **IFCSEG** operand, **SORT**s the entire file.

Example 3:

```
SORT INPCB(INVDB) OUTFILE(SYSUT2),
        IF(SEG,EQ,C'ITEM'),
        SORTPARM(SORT FIELDS(54,3,CH,A))
```

Copies only those INVDB database `ITEM` segments selected by the **IF** operand, and **SORT**s the selected segments.

Deleting Command

Command	Description
DELETE	Delete selected segments.

DELETE Command

Delete selected segments in a database.

Command Format

```
DELETE[operands...]
```

Consideration should be given as to whether the database can be recovered in the event that the **DELETE** command is unable to complete. It may be necessary to back up the database before processing the **DELETE** command.

When deleting, a SYNCPOINT is issued every 100 segments.

Examples of DELETE

Example 1:

```
DELETE INPCB(PODB),
            IF(SEG,EQ,C'ORDER'),
            ANDIF(30,LT,C'1990')
```

`ORDER` segments with dates prior to the year `1990` would be selected for processing and deleted from the database.

Option Command

Command	Description
OPTION	Set return codes and test conditions for any commands that follow in the same step.

OPTION

Set return codes and test conditions for commands that follow in the same step of the job. This command can be used multiple times to change the settings of the various return codes and conditions.

Command Format

```
OPTION {operands}
```

Examples of OPTION

Example 1:

```
OPTION EMPTYRC(4)
```

In the commands that follow, if the input file is found to be empty, set the return code for that command to `4`.

Note: This overrides both the MAXIBAT default set by the MAXDFLTS table (see "Appendix D: Changing Installation Defaults" on page 119 for further information).

Example 2:

```
OPTION SETMAXRC(0)
```

Will set the value of the maximum return code at this point in the command stream to zero. This will override any values set prior to this point in the command set.

Printing and Reporting Commands

All **PRINT** commands print segments horizontally and include segment and key information.

Command	Description
PRINT	Print selected segments.
PRINTALL	Print all segments.
PRINTIMS	Print selected database records.

PRINT Command

Print segments horizontally including segment & key information.

Command Format

{PRINT|PRINTALL|PRINTIMS} [operands ...]

PRINT prints segments in a horizontal format from an input database or file. Information about the segment such as segment name, segment key and segment key length are included for each segment on the report.

The **PRINT** command can be used to produce a formatted report of expanded copybook information. See "*Appendix C: Copybook Support*" on page 117 for details on this feature.

The report is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT.

```
MAX PRINT SERVICES                              DATE=22 Dec 1999 TIME=12:37:22                        PAGE 1 SEGMENT NAME = PARTROOT
KEY LENGTH=17 KEY POSITION=1
 OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
      0 F0F2C1D5 F9F6F0C3 F1F34040 40404040 40                                    *02AN960C13                      *
 SEGMENT - LENGTH=50
 ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0----+---11----+---12----+---13
 02AN960C13               WASHER

 SEGMENT NAME = STANINFO       KEY LENGTH=2 KEY POSITION=1
 OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
      0 F0F2                                                                      *02                              *
 SEGMENT - LENGTH=85
 ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0----+---11----+---12----+---13
 02               742                        1201  15      A6C      06C

 SEGMENT NAME = STOKSTAT       KEY LENGTH=16 KEY POSITION=1
 OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
      0 F0F040C1 C1F1F6F5 F1F14040 40404040                                       *00 AA16511                      *
 SEGMENT - LENGTH=160
 ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0----+---11----+---12----+---13
 00 AA16511          000000000     EACH  0000000110000000                 512    0000000{0000131{0000015{0000020{0000126{0000104{0
 00000000000000{0000000{0 Z512N                                                                                                **
```

Figure 6: Sample **PRINT** Report

Examples of PRINT

Example 1:

```
PRINTALL  INPCB(PCB01)
```

Print all segments from database PCB01.

Example 2:

```
PRINT  INFILE(PCBOUT),
           IF(SEG,EQ,C'BACKORDR')
```

Print selected segments from an unloaded file containing data from PCBO1. Choose only segments named 'BACKORDR'.

Example 3:

```
PRINTIMS  INPCB(PCB01),
           IF(SEG,EQ,C'BACKORDR')
```

Print entire database records (all segments) if the database record contains a BACKORDR segment

All **FPRINT** commands print segments utilizing a COBOL/PL/I layout to map the data record. The report includes segment information and key.

Command	Description
FPRINT	Print formatted selected segments.
FPRINTALL	Print formatted all segments.
FPRINTIMS	Print formatted selected database records.

FPRINT Command

Print segments using segment layout. See also DUMP, LIST and PRINT.

Command Format

{FPRINT|FPRINTALL|FPRINTIMS} [operands ...]

FPRINT prints segments from an input database or unloaded file. The report produced will use a COBOL/PL/I record layout to map the data to field names. The data set that contains the record layout is identified by using one of the following techniques:

- Include the command operand **COPYBOOK(...)** for a COBOL layout.
- Include the command operand **PLICOPY(...)** for a PL/I layout.
- Include the command operand **MAPDD(...)** for mapping criteria.

See the appropriate command operand for each operand mentioned above for further information.

If none of the layout identifying command operands (**COPYBOOK(..)**, **PLICOPY(..)**, **MAPDD( )** are supplied the printed segments will be in **DUMP** format.

The formatted report is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT.

```
MAX PRINT SERVICES                              DATE=22 Dec 1999 TIME=12:35:16                      PAGE 1 SEGMENT NAME = PARTROOT
KEY LENGTH=17 KEY POSITION=1
 OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
      0 F0F2C1D5 F9F6F0C3 F1F34040 40404040 40                                    *02AN960C13                      *
 SEGMENT - LENGTH=50                  COPYBOOK DESC=PART DEFINITION
 POSS  *----------FIELD NAME---------* FORMAT *---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8---*
 00001 U                              C    2 02
 00003 ROOT-KEY                       C   15 AN960C13
 00027 ROOT-DESCR                     C   20 WASHER
 SEGMENT NAME = STANINFO       KEY LENGTH=2 KEY POSITION=1
 OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
      0 F0F2                                                                     *02                              *
 SEGMENT - LENGTH=85                  COPYBOOK DESC=STANDARD INFO_<=25
 POSS  *----------FIELD NAME---------* FORMAT *---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8---*
 00001 V                              C    2 02
 00019 STAN-PROC-CODE                 C    2 74
 00021 INVENTORY-CODE                 C    1 2
 00022 PLANNING-REVISION-NUMBER       C    2
 00048 MAKE-DEPT                      C    2 12
 00050 MAKE-COST-CTR                  C    2 01
 00054 COMMODITY-CODE                 C    4 15
 00062 MAKE-SPAN                      Z    3 X'C1F6C3'
```

Figure 7: Sample **FPRINT** report

Examples of FPRINT

Example 1

```
FPRINTALL INPCB(PCB01),
          MAPDD(IMSMAP)
```

Print formatted all segments from input database PCB01

Example 2:

```
FPRINT INPCB(PCB01),
       COPYBOOK (MXS.COPYLIB(BKSEQ)),
       IF(SEG,EQ,C'BACKORDR')
```

Print formatted selected segments from PCB01. Print only BACKORDR segments.

All **LIST** commands print segments horizontally as do the **PRINT** commands. However, segment and key information are not included.

Command	Description
LIST	List selected segments.
LISTALL	List all segments.
LISTIMS	List selected database records.

LIST Command

Print segments horizontally without segment or key information. See also FPRINT Command, DUMP Command and PRINT Command.

Command Format

{LIST|LISTALL|LISTIMS} [operands ...]

LIST prints segments from an input database or unloaded file. The report produced will be a horizontal printing of each segment. Segment name precedes one or more segments of the same type.

The list report is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT.

```
MAX PRINT SERVICES                                DATE=22 Dec 1999 TIME=12:09:09                    PAGE 1
 SEGMENT - PARTROOT
 02AN960C13               WASHER
 SEGMENT - STANINFO
 02                742                          1201  15      A6C      06C
 SEGMENT - STOKSTAT
 00 AA16511          000000000     EACH  0000000110000000                  512
 0000000{0000131{0000015{0000020{0000126{0000104{0
 00000000000000{0000000{0 Z512N
 00 AA16512          000000000     EACH  0000000110000000                  512
 0000000{0000131{0000015{0000020{0000126{0000104{0
 00000000000000{0000000{0 Z512N
 00 AK2877F          M000100000    EACH0000000000000270000                 360
 0000000{0000088{0000000{0000000{00000880000003700
 00000000000000{0000000{0 Z360N
 0028009126          000000000     EACH
 00000000000000000000000513517517S000000000000630{0000000{0000015{0000680{0001053{0
 000104{0000000000000000{0  494Y
 SEGMENT - PARTROOT
 02AN960C17               WASHER
```

Figure 8: Sample **LIST** Report

Examples of LIST

Example 1:

List all segments from input database PCB01.

```
LISTALL  INPCB(PCB01)
```

Example 2:

List entire database records from PCB01 that contain the year 1999 anywhere within the account segment.

```
LISTIMS INPCB(PCB01),
            IF(SEG,EQ,C'ACCTNG')  AND(1,0,C'1999')
```

All **DUMP** commands print segments in a dump style showing both the character and hexadecimal representation. Segment and key information are included.

Command	Description
DUMP	Dump selected segments.
DUMPALL	Dump all segments.
DUMPIMS	Dump selected database records.

DUMP Command

Print segments in a dump format. See also FPRINT Command, PRINT Command and LIST Command.

Command Format:

{DUMP|DUMPALL|DUMPIMS} [operands ...]

DUMP prints segments from an input database or unloaded file. The report produced will be a dump style of the segments. The dump report is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT.

```
MAX PRINT SERVICES                                DATE=22 Dec 1999 TIME=12:32:27                          PAGE 1
SEGMENT NAME = PARTROOT       KEY LENGTH=17 KEY POSITION=1
 OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
      0 F0F2C1D5 F9F6F0C3 F1F34040 40404040 40                                    *02AN960C13                      *
SEGMENT - LENGTH=50
 OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
      0 F0F2C1D5 F9F6F0C3 F1F34040 40404040 40404040 40404040 4040E6C1 E2C8C5D9 *02AN960C13                WASHER*
     32 40404040 40404040 40404040 40404040 4040                                 *                                *
SEGMENT NAME = STANINFO       KEY LENGTH=2 KEY POSITION=1
 OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
      0 F0F2                                                                     *02                              *
SEGMENT - LENGTH=85
 OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
      0 F0F24040 40404040 40404040 40404040 4040F7F4 F2404040 40404040 40404040 *02                742           *
     32 40404040 40404040 40404040 404040F1 F2F0F140 40F1F540 40404040 40C1F6C3 *               1201  15       A6C*
     64 40404040 4040F0F6 C3404040 40404040 40404040 40                         *      06C                       *
SEGMENT NAME = STOKSTAT       KEY LENGTH=16 KEY POSITION=1
 OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
      0 F0F040C1 C1F1F6F5 F1F14040 40404040                                      *00 AA16511                      *
SEGMENT - LENGTH=160
 OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
      0 F0F040C1 C1F1F6F5 F1F14040 40404040 40404040 F0F0F0F0 F0F0F0F0 F0404040 *00 AA16511           000000000   *
     32 4040C5C1 C3C84040 F0F0F0F0 F0F0F0F1 F1F0F0F0 F0F0F0F0 40404040 40404040 *  EACH  0000000110000000        *
     64 40404040 40404040 4040F5F1 F2404040 40F0F0F0 F0F0F0F0 C0F0F0F0 F0F1F3F1 *          512    0000000{0000131*
     96 C0F0F0F0 F0F0F1F5 C0F0F0F0 F0F0F2F0 C0F0F0F0 F0F1F2F6 C0F0F0F0 F0F1F0F4 *{0000015{0000020{0000126{0000104*
    128 C0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0 C0F0F0F0 F0F0F0F0 C0F040E9 F5F1F2D5 *{0000000000000000{0000000{0 Z512N*
```

Figure 9: Sample **DUMP** Report

Examples of DUMP

Example 1:

```
DUMPALL  INPCB (PCB01) .
```

Dump all segments from input database PCB01.

Example 2:

```
DUMPIMS   INPCB(PCB01),
            IF(SEG,EQ,C'STOKSTAT')  AND(25,EQ,C'ACT')
```

Dump selected segments from PCB01. Choose `STOKSTAT` segments with a status of active. Active records contain '`ACT`' beginning in segment position 25. Dump the entire database record.

Updating Commands

Command	Description
UPDATE	Update selected segments.

UPDATE Command

Update a database. Update selected segments in a database.

Command Format

```
UPDATE[operands ...]
```

Consideration should be given as to whether the database can be recovered in the event the **UPDATE** command was unable to complete. It may be necessary to backup the database before performing an **UPDATE** command.

When updating, a SYNCPOINT is issued every 100 segments that are changed.

Examples of UPDATE

Example 1:

```
UPDATE  INPCB(PCB01) ,
            IF(SEG,EQ,C'PARTROOT'),
            AND(15,EQ,C'01839')  REPL(15,C'01792')
```

Update Selected records in database PCB01. The selected segments named 'PARTROOT' which contain '01839' in position 15 are changed to '01792'.

Example 2:

```
UPDATE  INPCB(PCB01),
            IF(SEG,EQ,C'BACKORDR') CHNG(25,EQ,C'CHICAGO',C'NEWYORK')
```

Update all BACKORDR segments that were originated in Chicago to New York.

CHAPTER 4: COMMAND OPERANDS

Operand Summary

The following is an alphabetical list of MAX IMS/UTIL Batch command operands.

Operands are used either to specify global settings or control the processing of segments by a command.

Each segment will be processed by operands in the order the operands are specified.

Operands may be coded in a sequence such that precise segments may be selected for processing by subsequent operands. For example, an **IF** operand placed before a **PRINT** operand would ensure that certain segments are not printed until they have been selected by the **IF** operand.

Segment Selection Operands

Operand	Description
AND	Select specific segments or database records in combination with a preceding IF.
COMPARE	Select column ranges to compare.
FREQOUT	Specify output selection frequency.
IF	Select specific segments or database records.
IN	Maximum number of segments or database records input.
LIMIT	Maximum number of selected segments or database records to output.
OR	Select Specific segments or database records in combination with a preceding IF.
MATCH	Select records based upon data in another file.
MATCHKEY	Select database record based on specific root key in the MTCHFILE file.
OUT	Maximum number of segments or database records to output
SELECT	Select every nth segment or database record.
SKIP	Skip further processing of this segment or database record.
STOP	Stop processing at selected segment or database record.

Segment Manipulation Operands

Operand	Description
CALCAMT	Re-calculate a numeric field.
CALCDATE	Re-calculate a date field.
CHANGE	Change a segment in data file mode.
DELETE	Delete a segment from the database.
EDIT	Change a segment in text file mode.
MOVE	Build a segment.
REPLACE	Overlay data in a segment.
SCRAMBLE	Encode data in a segment.
TRANSLATE	Change data in a segment to a new set of values.
UNSCRAMBLE	Decode data in a segment.

Printing Operands

Operand	Description
DUMP	Dump segment with segment information and record key.
FPRINT	Print segment formatted using record layout.
LIST	Print segment.
PRINT	Print segment with segment information and segment key.

File Name Operands

Operand	Description
COMPFILE	Compare to file name.
COMPPCB	Compare input from a PCB.
INFILE	Read input from file name.
INPCB	Read primary input from a PCB.
MTCHFILE	Match data from file name.
OUTFILE	Write output to file name.
OUTPATH	Direct the output of a **COPY** operation to a UNIX file.
OUTPCB	Write output to a PCB.
WRITE	Write specific selected segment to file name.

Initial Segment Key Positioning Operands

Operand	Description
COMPRID	Initial segment key positioning for database or VSAM/SAM file that is the COMPFILE for COMPARE.
INRID	Initial segment key positioning for database or VSAM/SAM file that is the INFILE for COMPARE.
RID	Initial segment key positioning for database or VSAM/SAM file types.

Miscellaneous Operands

Operand	Description
ABEND	Abend when there is a processing error.
ACCUM	Accumulate and print the total value for a field.
CODEPAGEIN	Specify code page for input data being transformed using the FORMAT operand.
CODEPAGEOUT	Specify code page for output data being transformed using the FORMAT operand.
COMPLIM	Dynamically alter the match limit for the re-synchronization of a COMPARE.
FORM	Control print format, output format, and/or establish processing options.
FORMAT	Control the format of data being output to a data set or UNIX file.
GDATES	Specify date range in Gregorian for a CALCDAYS.
JDATES	Specify date range in Julian for a CALCDAYS.
PADCHAR	Filler character used for padding incomplete segments.
PIN	Specify a value to use when encoding and decoding data.
SETRC	Set specific return code.
SORTPARM	Pass parameters to the **SORT** program.

Formatted Processing Operands

Operand	Description
COPYBOOK	COBOL layout data set name.
MAPDD	File name of mapping criteria data set.
PLICOPY	PL/I layout data set name.
REFDD	File name of reformatting criteria data set.

Option Statement Operands

Operand	Description
CBNFRC	Override copybook not found on FPRINT command.
COMPRC	Override return code when not all records match during the COMPARE command.
EMPTYRC	Override the return code for an empty input file (INFILE) or database (INPCB).
MATCHKEYRC	Override the return code when all the keys (MATCHKEY) are not found on the INFILE or INPCB.
NOSELRC	Override the return code when the selection operands (IF, AND, OR) result in no segments being selected from the INFILE or INPCB.
SETMAXRC	Set the maximum return code value.
SETRC	Set the return code value.
STOPMAXRC	Bypass any further commands in that step when the maximum return code meets specified condition.
STOPRC	Bypass any further commands in that step when a return code meets specified condition.
TRANSRC	Override the return code if transformation errors occur as the result of a missing copybook.
TRUNCRC	Override the return code if record truncation occurs.
WARNMATCHKEY	Set warning on/off and set limit for mismatches on MATCHKEY vs. INFILE or INPCB.
WARNTRANERR	Set warning on/off and set limit for errors during data transformation.

ABEND

Use **ABEND** to force an abnormal end for processing error. If processing an IMS database, an abend is not issued but processing is stopped with a Return Code `16`.

This operand should only be specified once per command.

Syntax:

ABEND({0|1|2|3})

Sub-operands:

0	Do not abend for any processing errors.
1	Abend when I/O problem detected.
2	Abend for any processing problem with `RC>8`.
3	Abend if **VERIFY MODE** ends with `RC>8`, or if **PROCESSING MODE** ends with `RC>4`.

ACCUM

Accumulate and print the totals for a specific field in selected segments.

If the operand is used with the **COMPARE** command, totals will be accumulated for each of the inputs. Upon completion of the **COMPARE**, the totals from each input will be compared and printed. Totals that do not match will result in a return code =4.

Syntax:

```
          ACCUM(position[,'description'])
or        ACCUM(position,length,data-type[,'description'])
```

Sub-operands:

position	Position in segment.	
	1-32760	The actual position number
	+nnn or -nnn	The relative position from the last selected position.
length	Length of field	When 'length' and 'data-type' sub-operands are omitted, the data-type for the field must be packed.
data-type	C	Zoned decimal with a maximum length 15.
	B	Binary halfword or fullword.
	P	Length sub-operand is ignored.
	R	Report type data: this data-type allows you to accumulate totals that have been edited into report format. This will accumulate up to 999 billion and allows 10 decimal positions.
description	Description to appear next to accumulated totals in report.	

Example:

ACCUM(10,6,C,C'ZONED VALUE')

Will accumulate a total for a six position zoned decimal field starting in position 10. The total will be given the title of 'ZONED VALUE'.

ACCUM(25,,P)

Will accumulate a total for packed data beginning in position 25. As no description is provided, the total will be identified by the beginning position number.

CALCAMT

Re-calculate an amount field in a segment.

This operand can be used to re-calculate values. In addition, it can be used to change the format of the data. `Position1` and `format1` always refer to the first value to be used in the calculation. The second value can either be hard coded or taken from the data record. The result can replace the original value and format, or can be returned in a different `position3` and/or `format3`.

Syntax:

#1	**CALCAMT(position1,format1,function,value)**
#2 or,	**CALCAMT(position1,format1,function,,position2,format2)**
#3 or,	**CALCAMT(position1,format1,function,,position2,format2,position3)**
#4 or,	**CALCAMT(position1,format1,function,,position2,format2,position3,format3)**
#5 or,	**CALCAMT(position1,format1,function,value,,,position3,format3)**

Sub-operands:

position	Position in segment.	
	`1-32760`	The actual position number.
	`+nnn` or `-nnn`	The relative position from the last selected position.
format*n*	Format of data at position '`n`'.	
	Valid formats for decimal data:	
	`Pn.n`	Packed signed.
	`Un.n`	Packed unsigned.
	`Zn.n`	Zoned with a sign.
	`Nn.n`	Zoned with no sign.
	`n.n`	Indicates number of digits before and after an assumed decimal position. This is necessary to maintain decimal alignment in result. If no decimal digits follow decimal point, only the first '`n`' needs to be coded.
	Valid formats for binary data are:	
	`H`	Halfword, 2 bytes.
	`F`	Fullword, 4 bytes.
	`D`	Doubleword, 8 bytes.
functions	*	Multiply.
	+	Add to.
	-	Subtract from.
	/	Divide by.

value	+ nnn	A positive value for adjustment.
	- nnn	A negative value for adjustment.
Format#1:	Value to use in calculation is 'hard coded' in the operand as the 4th sub-parameter. Result will be returned in position 1 from the parameter list using `format1`.	
Format#2:	Value to use in calculation is extracted from the record using `position2`, `format2`. Result will be returned in `position1` from the parameter list using `format1`.	
Format#3:	Value to use in calculation is extracted from the record using `position2`, `format2`. Result will be returned in `position3` from the parameter list using `format1`.	
Format#4:	Value to use in calculation is extracted from the record using `position2`, `format2`. Result will be returned in `position3` from the parameter list using `format3`.	
Format#5:	Value to use in calculation is 'hard-coded' in the operand as the 4th sub-parameter. Result will be returned in `position3` using `format3`.	

Examples of CALCAMT

Example 1:

CALCAMT(25,U7,*,,20,H)

Re-calculate a value using data from two fields within the segment, return result in field and format of the first value. Field one is packed, unsigned, with no decimal digits and at position `25` within the segment. Field two is a binary halfword and at position `20` within the segment. (Use Format #2.)

Example 2:

CALCAMT(20,P5.2,*,1.5)

Re-calculate a value by multiplying it by `1.5`, return to field one. Field one data is packed, signed and contains 2 decimal digits and at position `20` within the segment. (Use Format #1.)

Example 3:

CALCAMT(20,P5.2,*,1.5,,,20,P4.3)

Prior example, if the result should contain 3 decimal digits, use `position3` and `format3` to return it to original location with new format. (Use Format #5.)

CALCDATE

Calculate a number of days plus/minus for a specific field in selected segments.

When this operand is used with the **COMPARE** command, it is used to manipulate dates in the **INFILE** only. If the **COMPFILE** has had its dates aged with a prior process, dates in the **INFILE** can be aged during the **COMPARE**, forcing them to compare equal. This eliminates that field as a possible mismatch.

Syntax:

CALCDATE (position,format,num_days[,exit_pgm][,edit_mask])

Sub-operands:

position	Position in segment.	
	1-32760	The actual position number.
format	The following format codes describe the date fields. {J1|J2|G1|G2|G3|EM}[P|Z][UN|SI][SE|NS] [NC|CE][X] The first code provides the format of the date or indicates that an edit mask will provide the format. This is always the first code and is required. J1 (yyddd) J2 (ddyyy) G1 (yymmdd) G2 (mmddyy) G3 (ddmmyy) EM (edit mask provided) The remaining codes are optional, can be provided in any order and further define the field format.	
	Indicate data format, valid with all field types and edit mask.	
	P	Packed
	Z	Zoned. This is Default.
	Indicate presence or absence of a sign. Valid with all field types and edit mask.	
	SI	Signed
	UN	Unsigned. This is Default.
	Further describe the date, not valid with EM (edit mask) format.	
	SE	Separator character included, valid with zoned decimal.
	NS	No Separator, valid with zoned decimal. This is Default.
	CE	Century included.
	NC	No century included. This is Default.
	Use to indicate date to be passed to an exit program.	
	X	Exit program provided.
num_days	+nnn or -nnn	A number of days forward or backwards.
	*	Use of an asterisk in this field signals the operand to use the number of day as calculated by the **CALCDAYS** command.
exit_pgm	Name of a user written exit program called after date has been calculated. See member "MAXPDXIT" in the installation JCL library for details on coding this user exit program. This exit may be used to process specialized date logic such as an accounting period date.	

edit_mask	If the format contains the character EM indicating that one of the standard date formats is not being used, an edit mask is used to describe the date field.
	Valid mask characters are: M (month); D (day); Y (year); C (century); / (separator)
	Note: Separator does not have to be a slash, but a slash is used to signal a separator present in that position.
	Example edit mask : A date in format of month, year, day would be represented with a mask of 'MMYYDD' if the century is present specify as 'MMCCYYDD'.

Examples of CALCDATE

Example 1:

CALCDATE(10,G1ZSECE,+45)

Add 45 days to the date in position 10 with a format of 'YYYY/MM/DD'.

G1	**(yymmdd)**
Z	**(Zoned)**
SE	**(Separator character)**
CE	**(Century included)**

Example 2:

CALCDATE (10,EMP,*,,MMCCYY)

Add number of days computed by **CALCDAYS** command to the date in position 10 as described by the edit mask.

EM	**Edit mask**
P	**Packed date**
MMCCYY	**Date contains month and four-digit year, no day.**

CBNFRC

Set the value for the return code from an **FPRINT** command that follows if the copybook needed to format the record is not found. This value overrides both the MAXIBAT default and the installation default for this step only. See "Appendix D: Changing Installation Defaults" on page 119.

Syntax:

CBNFRC(n)

Sub-operands:

n	Numeric value for return code.

Example:

CBNFRC(8)

Would set the command return code to `8` if all copybooks for an **FPRINT** command were not found.

CHANGE

Changes data in a segment.

When the lengths of the **to-data** and **from-data** are not equal, the data on the right side of the found **to-data** will be shifted left/right to accommodate the extra/missing bytes.

When processing an **UPDATE** command, the fixed length segment length cannot change. Therefore, if the segment length does change as a result of a **CHANGE** operand, upon rewriting, the segment will be either truncated or padded (see **PADCHAR** operand on page 81) to its original length. A count of truncated records is provided.

When processing a **COPY** command to unload or copy segments to a sequential file, record lengths will be determined by the LRECL and RECFM of the output file. Variable records will be written reflecting their true size up to a maximum length of the specified LRECL. Fixed length records will be padded (see **PADCHAR** operand on page 81) or truncated to equal the LRECL. A count of truncated records is provided.

See **COPY with CHANGE** example on page 101.

Syntax:

CHANGE(position,length,[dupl]from-data,[dupl]to-data)
or, CHANGE(position,operator, [dupl]from-data,[dupl]to-data)
or*, CHANGEALL(position,length, [dupl]from-data,[dupl]to-data)

***CHANGEALL** is used to change every occurrence of the **from-data** found within the specified length for the segment.

Sub-operands:

position	Position in segment.	
	`1-32760`	The actual position number.
	`+nnn` or `-nnn`	The relative position from the last selected position.
length	Length to scan.	
OR	`0`	Rest of segment.
operator	`EQ`	Equal.
	`NE`	Not equal.
	`GT`	Greater than.
	`GE`	Greater equal.
	`LE`	Less than or equal.
	`LT`	Less than.
dupl	Number of repeating occurrences of data (`C"..."`, `X"..."`, `P"..."`). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	
from-data	`C"x"`	Character case sensitive.
	`C""`	Any character.
	`C'xx[,xx, ...]'`	Separate character strings with '`OR`' condition to be used for processing each string.
	`X"hh"`	Hexadecimal.
	`P"nn"`	Packed, all leading zeroes should be specified.
	`T"x"`	Case insensitive.
to-data	`C"x"`	Character.
	`C""`	Null.
	`X"hh"`	Hexadecimal.
	`P"nn"`	Packed, all leading zeroes should be specified.

CODEPAGEIN

Specify the code page of the file used as input to data transformation using the **FORMAT** operand. If no input code page is specified, the code page of 37, EBCDIC, U.S. English is used.

This operand should be coded only once per command. During the verify process, this will be validated to be a supported code page.

Syntax:

CODEPAGEIN(nnnnn)
CPI(nnnnn)

Sub-operands:

nnnnn	Code page value.

Example:

CPI(1140)

Input to transformation is using code page `1140`, EBCDIC, U.S. English with Euro.

CODEPAGEOUT

Specify the code page to be used during data transformation when writing data using the **FORMAT** operand. If no output code page is specified, either the input code page is used if specified, or the default code page for the selected format is used (XML, TAB, CSV).

This operand should only be used once per command. During the verify process, this will be validated to be a supported code page.

Syntax:

CODEPAGEOUT(nnnnn)
CPO(nnnnn)

Sub-operands:

nnnnn	Code page value.

Example:

CPO(1140)

Will transform data to output using code page `1140`, EBCDIC, U.S. English with Euro.

COMPARE

Compare specific positions from segments in the **INPCB** or **INFILE** to specific positions in the **COMPFILE** or **COMPPCB**.

This operand may be specified as many times as necessary to compare all required positions. Use this operand to compare segments in files that have different record lengths or when fields have been rearranged, or to exclude certain positions from being compared. Use the second format of this operand to compare data for a specific value.

Syntax:

COMPARE(in-position,length,comp-position)
or, COMPARE(in-position,[dupl]data)

Sub-operands:

in-position	Position in segment of the input data to compare.	
	`1-32760`	The actual position number
length	Length of data to compare.	
	`0`	Rest of segment.
comp-position	Position in segment of the compare data to compare.	
	`1-32760`	The actual position number.
dupl	Number of repeating occurrences of data.	
data	`C"x"`	Character case sensitive.
	`X"hh"`	Hexadecimal.
	`P"nn"`	Packed, all leading zeroes should be specified.

COMPFILE

Specify the compare to file name.

Use this operand when the compare type of file is an unloaded copy of the database. Use this when the compare file is either SAM or VSAM. This operand may be coded once per command and cannot be coded in combination with the **COMPPCB** operand. No default value is provided for this operand. Reserved ddnames as listed in the Allocations table on page 6 should not be used.

Syntax:

COMPFILE(ddname) Compare only

Sub-operands:

ddname	Filename of compare to data set.

COMPLIM

Control the number of segments or records that are compared before issuing a mismatch condition.

This limit is used to direct the **COMPARE** to read forward in the opposite file, looking at each segment or record for a match. If no match is found within this limit, a mismatch is declared and processing begins with the next record in each set of input.

Syntax:

COMPLIM(number)

Sub-operands:

number	Number of segments to search (default = 50).

COMPPCB

Specify the compare to database PCB name contained in the PSB coded with the DFSRRC00 parameter list.

This operand should only be specified once per command and cannot be coded in combination with the **COMPFILE** operand. If a **COMPARE** command is coded with neither a **COMPFILE** nor **COMPPCB** operand, the default is **COMPPCB(#1)**.

Syntax:

COMPPCB (pcbname|#pcbnumber)

Sub-operands:

pcbname	Database PCB name of primary input.
#pcbnumber	The relative position number of the database PCB within the PSB. Precede the number with the number sign,#, Example #2. Non-database PCBs are not included when determining the relative position number.

COMPRC

Set the return code value for any **COMPARE** commands that follow in the same step where all records being compared do not match. This value overrides both the MAXIBAT default and any installation default for this step only. See "Appendix D: Changing Installation Defaults" on page 119.

Syntax:

COMPRC(n)

Sub-operands:

n	Numeric value for return code.

Example:

COMPRC(0)

Would set the command return code to zero for any **COMPARE** command even if segments were found not to match (inserts, deletes, mismatches).

COMPRID

COMPRID positions the **COMPPCB** or the **COMPFILE** (in IMS format) to a specific root segment.

If the **COMPFILE** is not in IMS format, see chart below for **RID** value.

This operand should only be specified once per command.

This operand is only valid with the **COMPARE** command.

Syntax:

COMPRID(rid | root segment key)

Sub-operands:

rid	For ESDS this is a 4 byte (`X'hhhhhhhh'`) hexadecimal RBA.
	For KSDS, this is a character, or hexadecimal, full or partial key.
	For RRDS, this is a numeric relative record number.
	For SAM this is a numeric actual record number.
root segment key	Root key for database or unloaded file in IMS format.

COPYBOOK

COPYBOOK identifies the data set and member name of the COBOL copybook to be used for mapping the segments with the formatted print features (see also: **COMPARE** and **FPRINT** commands and **FPRINT**( ) operand).

COPYBOOK( ), **MAPDD**( ), **PLICOPY**( ) are mutually exclusive, choose only one.

This operand should only be specified once per command.

Syntax:

COPYBOOK(dsn(mem))

Sub-operands:

dsn(mem)	Fully qualified data set and member name to a COBOL copybook.

DELETE

Delete selected segments.

This operand is valid only with the **UPDATE** command. Segments that are selected for processing are deleted from the database.

Consideration should be given as to whether the database can be recovered in the event that the command is unable to complete. It may be necessary to back up the database before processing with the **DELETE** operand.

Records cannot be deleted from an ESDS VSAM file being updated in place.

Syntax:

DELETE

Note: The **DELETE** operand has no sub-operands.

Example:

```
IF(SEG,EQ,C'ORDER'),
ANDIF(1,EQ,C'OLDREC') DELETE
```

If the value 'OLDREC' was found in positions 1 through 6, the ORDER segment would be deleted.

DUMP

Print selected segments.

Segments are printed in a dump style format. To print records from an **INFILE** that is not in IMS format use **FORM(NOIMS)**.

The report is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT.

Syntax:

DUMP(number)

Sub-operands:

0	No limit on number of segments printed.
number	Limit the number of segments printed.

EDIT

Use **EDIT** to change data in a text type field.

When the length of the **to-data** is greater than the **from-data** length text will be shifted right and any repeating space characters following the **from-data** plus connected text will be removed. If necessary, the record length may be increased to accommodate the new data. If the output file's RECFM or LRECL do not permit the larger record length, the record may be truncated.

When the length of the **to-data** is less than the length of the **from-data**, the **PADCHAR()** operand character will be used to fill the void after the first space character is located to the right of the **from-data**.

When processing an **UPDATE** command, the segment length cannot change. Therefore, if the segment length does change as a result of an **EDIT** operand, upon rewriting, the segment will be either truncated or padded (see **PADCHAR** operand on page 81) to its original length. A count of truncated records is provided.

EDITALL is used to change every occurrence of the **from-data** found within the specified length for the segment.

Syntax:

EDIT(position,length,[dupl]from-data,[dupl]to-data)
or, EDIT(position,operator, [dupl]from-data,[dupl]to-data)
or, EDITALL(position,length, [dupl]from-data,[dupl]to-data)

Sub-operands:

position	Position in segment.	
	`1-32760`	The actual position number.
	`+nnn` or `-nnn`	The relative position from the last selected position.
length	Length to scan.	
OR	`0`	Rest of segment.
operator	`EQ`	Equal.
	`NE`	Not equal.
	`GT`	Greater than.
	`GE`	Greater equal.
	`LE`	Less than or equal.
	`LT`	Less than.
dupl	Number of repeating occurrences of data (`C"..."`,`X"..."`,`P"..."`). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	
from-data	`C"x"`	Character case sensitive.
	`C'xx[,xx, ...]'`	Separate character strings with '`OR`' condition to be used for processing each string.
	`X"hh"`	Hexadecimal.
	`P"nn"`	Packed, all leading zeroes should be specified.
	`T"x"`	Case insensitive.
to-data	`C"x"`	Character.
	`C""`	Null.
	`X"hh"`	Hexadecimal.
	`P"nn"`	Packed, all leading zeroes should be specified.

EMPTYRC

Set the return code value for any command that follows where the **INFILE** or **INPCB** is found to be empty. This value overrides both the MAXIBAT default and any installation default for this step only. See "Appendix D: Changing Installation Defaults" on page 119.

Syntax:

EMPTYRC(n)

Sub-operands:

n	Numeric value for return code.

Example:

EMPTYRC(12)

Would set the command return code to `12` for any command that follows in the same step where the **INFILE** or **INPCB** was found to be empty.

FORM

Control various processing and printing options.

Each variation of this operand should only be specified once per command.

Syntax:

FORM ({LONG|SHORT})
or, FORM(IMS|IMSCSEG|IMSCKEY|NOIMS)
or, FORM({MULTI|NOMULTI})
or, FORM({[C|F|H],[D|S|L]}) COMPARE/COMPAREALL command only
or, FORM({ORIG|SUBREC}) COMPARE/COMPAREALL command only

Sub-operands:

IMS	Data in the input file, output file or compare to file is in MAX IMS/UTIL Batch unloaded IMS FORMAT. Use of this operand allows use of the SEG format of the **IF, AND, OR**, or **STOP** operands. In addition, position sub-operands reflect true positions within the segment. Knowledge of the unloaded file format is not required. (DEFAULT)
NOIMS	Data in **INFILE**, **OUTFILE** or **COMPFILE** is not an unloaded IMS database. Use this operand to copy segments with no control prefix or to ignore the control prefix.
IMSCSEG	Data in the output file is in MAX IMS/UTIL Batch unloaded IMSCSEG format. This format includes the standard IMS prefix (16 bytes), the IMS concatenated key (255 bytes), and the segment data. When segments with this format are loaded back into the database, each one is inserted using the embedded concatenated key.

IMSCKEY	Data in the output file is in MAX IMS/UTIL Batch unloaded IMSCKEY format. This format includes the standard IMS prefix (16 bytes) followed by the IMS concatenated key (255 bytes). Segments may not be loaded back into the database as there is no segment data.
LONG	Include column bar for every record printed. For formatted **COMPARE** output, include every field in the report. (DEFAULT)
SHORT	Include column bar only at top/bottom of page. For formatted **COMPARE**, include only fields with differences.
MULTI	Do not close the input file or PCB or change record position between commands.
NOMULTI	Close input file or PCB and position to first record or segment between commands. (DEFAULT)
CLOSEIN	Close the input file or PCB and position to first record between commands.
NOCLOSEIN	Do not close the input file or PCB, or change record position between commands.
CLOSEOUT	Close output file between commands.
NOCLOSEOUT	Do not close the output file or change record position between commands.
HEX	Print in character with hex characters below each character (3-line format). This format forces the **FORM(SHORT)** option.
The following operands are valid for the **COMPARE** command only.	
C	Horizontal style report.
F	Formatted report, using "records" layouts. When a formatting operand (**COPYBOOK**, **PLICOPY**, and **MAPDD**) is supplied, the formatted style report will become the default.
H	Dump style report. This is the DEFAULT when no formatting operands are specified.

D	Show differences and a summary. In formatted mode, all the fields are shown on the left, and only fields with differences are shown on the right. To print differing fields only, see **FORM(SHORT)** above. (DEFAULT)
L	Show all records noting differences and include summary.
S	Show summary only.

ORIG	Print the entire record that entered the **COMPARE** command. (DEFAULT)
SUBREC	Prints the segment as constructed for the **COMPARE** using the **COMPARE** operands showing differences only in the subset of the segment.

FORMAT

Use data transformation to format the output that is directed to a data set or UNIX file.

Segments are formatted using a record layout that has been identified using either a **COPYBOOK(..)**, **PLICOPY(..)** or **MAPDD(..)** operand. Default mapping criteria may also be used if copybook names as contained in a COPYLIBS data set are the same as the segment name. When a layout is not found to match, the segment is bypassed, as no data transformation can take place.

Syntax:

FORMAT({XML|TAB|CSV|other})

Sub-operands:

XML	Output is in XML format.
TAB	Output is tab delimited at each field.
CSV	Output is comma separated at each field value.

This operand is intended for use with the **COPY** command when the output is directed to an **OUTPATH**. However, it is possible to review the output by using a **LIST** command, and the data will be printed as it would appear in the output, breaking a line at each NL (new line) character. It is also possible to **COPY** the output to a data set that is defined as variable. However, the size of the record cannot exceed the maximum size of the records supported for the corresponding file type.

TAB and CSV formatted output will place the segment name as the first field in each line, while **LISTIMS** and **COPYIMS** commands will transform each database record into a single line.

The XML (eXtensible Markup Language), CSV (comma-separated variable), and TAB (tab character delimited) formats are provided. In addition, you may define your own format(s). There are sample macros defined in the MAXDFLTS member of the .JCL library. These examples provide models that will enable you to create your own (other) transformation formats. The format name should be 3-7 characters and cannot be the same as any predefined format.

Example:

FORMAT(XML)

Would write the data in XML format.

FPRINT

Print segments using copybook layout. Print selected fields.

Segments are printed using a segment layout that has been identified using either the **COPYBOOK(..)** **PLICOPY(..)**, or **MAPDD(..)** operands. When a layout is not available then the segments will print in a dump style format.

Use MAX IMS/UTIL to create mapping criteria. If no **MAPDD** is specified, default criteria is used where each segment's copybook is assumed to be in a member with the same name as the segment and located in a data set defined by the **COPYLIBS** ddname.

Mapping criteria can be used to select fields to be included in or excluded from the print, limiting the amount of data that has to be printed.

The report is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT. See **FPRINT** example on page 102.

Syntax:

FPRINT(number)

Sub-operands:

0	No limit on number of segments printed.
number	Limit the number of segments printed.

FREQOUT

Used to control the frequency of selected segments or database records that are written to the output of a copy command, or printed as output of a print type command. When used with IMS-mode processing, initiated by use of one of the following commands

COPYIMS, PRINTIMS, LISTIMS, FPRINTIMS, DUMPIMS

the output frequency is applied to the entire database record. Once a record is selected to be processed, every nth record is included in the ouput as coded in the **FREQOUT(n)** operand. In non IMS-mode, the frequency applies to the segment. In this case, every nth selected segment will be included in the output.

Note: This frequency applies after selection criteria have caused the segment or database record to be chosen for processing.

This operand may be coded one time per command.

Syntax:

FREQOUT(number)

Sub-operands:

0	No frequency, include every record.
number	A whole number, indicates every nth selected record to be included.

GDATES

Calculate the number of days between the first and second dates.

This value can be either positive (if second date is greater) or negative (if second date is less). This value is then used in any subsequent **CALCDATE** operands where an asterisk is used in place of the **num_days** sub-operand.

This operand is only valid with the **CALCDAYS** command.

Syntax:

GDATES(yyyy/mm/dd,yyyy/mm/dd)

Sub-operands:

yyyy/mm/dd	Beginning Gregorian date.
yyyy/mm/dd	Ending Gregorian date.

IF, AND, OR

The **AND(..)** and **OR(..)** operands are used to logically connect selection criteria to the preceding **IF(..)**, this is known as an **IF-** sequence.

The **IF-** sequence is used to select *segments* for processing by the following commands:

ACCUM, **COMPARE**, **COPY**, **PRINT**, **FPRINT**, **LIST**, **DUMP**, **UPDATE**

The **IF-** sequence is used to select *entire database records* for processing by the following commands:

ACCUMIMS, **COPYIMS**, **PRINTIMS**, **FPRINTIMS**, **LISTIMS**, **DUMPIMS**

The **IF-** sequence is used to select *segments to not be processed* by the following commands:

SKIP

The **IF-** sequence is used to select *segments for additional processing* by the following commands:

ACCUMALL, **COMPAREALL**, **COPYALL**, **PRINTALL**, **FPRINTALL**, **LISTALL**, **DUMPALL**

When multiple selection criteria are specified within a single **IF** operand they are **OR**'ed together. When any single criteria passes the test against the data segment the entire **IF** is considered successful.

Note: Multiple selection criteria must be specified within single quotes.

Example of multiple selection criteria within a single **IF**:

```
IF(100,EQ,C'APRIL,MAY,JUNE')
```

This statement will look at position 100 and test to see if it is equal to one of the three values listed. If it is equal to any one of the three values, it is considered to be true.

An exception to the rule of '**OR**' ing the result of testing the **IF** operands is when the '**NE**' operator is used. In this case, the operands are **AND**ed together.

For example:

```
IF(100,NE,C'APRIL,MAY,JUNE')
```

causes the segment to be selected if position 100 was not equal to any one of the three values.

The operands that follow any of the **IF**s are subordinate to the **IF** until another **IF (..)**, **OR(..)**, or **AND(..)** operand is detected. When the test on the data to the **IF** criteria is successful, the subordinate operands will be processed.

When an **IF** operand immediately follows another **IF** or an **OR** without any operands in between, the second **IF** is treated logically as an '**AND**'.

The following is a list of the subordinate operands to **IF**:

ACCUM, CALCAMT, CALCDATE, CHANGE, COMPARE, DUMP, EDIT, FPRINT, IN, LIST, MOVE, OUT, PADCHAR, PRINT, REPLACE, SELECT, SETRC, SKIP, STOP, WRITE.

Syntax:

```
        IF(position,length,[dupl]data[,pos,len,data, . . .])
or,     IF(position,operator,[dupl]data[, . . .])
or,     IF(position,length,[dupl]data-type[, . . .])
or,     IF(SEG,operator,segname)

        AND(position,length,[dupl]data[,pos,len,data, . . .])
or,     AND(position,operator,[dupl]data[, . . .])
or,     AND(position,length,[dupl]data-type[, . . .])
or,     AND(SEG,operator,segname)

        OR(position,length,[dupl]data[,pos,len,data, . . .])
or,     OR(position,operator,[dupl]data[, . . .])
or,     OR(position,length,[dupl]data-type[, . . .])
or,     OR(SEG,operator,segname)
```

Note: The **SEG** option is valid for PCB and IMS format unloaded files only.

Sub-operands:

position	Position in segment.	
	1-32760	The actual position number.
	+nnn or -nnn	The relative position from the last selected position.
SEG	Indicates compare for a specific segment name.	
length	Length to scan	
OR	0	Scan rest of segment.
operator	EQ	Equal.
	NE	Not equal.
	GT	Greater than.
	GE	Greater equal.
	LE	Less than or equal.
	LT	Less than.
dupl	Number of repeating occurrences of data (C"...",X"...",P"..."). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	
data	C"x"	Single character string that is case sensitive.
	C'xx[,xx, ...]'	Multiple character strings, an 'OR' condition will be used for processing each string.
	X"hh"	Single hexadecimal string.
	X'xx[,xx, ...]'	Multiple hexadecimal strings, an 'OR' condition will be used for processing each string.
	P"nn"	Single Packed decimal string.
	P'xx[,xx, ...]'	Multiple Packed decimal strings, an 'OR' condition will be used for processing each string. **Note:** Packed data need not contain leading zeroes. A search is made for a valid packed field of 1 to 19 digits. Then a packed compare is performed. Signs of 'F' or 'C' will be compared as equal and are considered positive. A sign of 'D' is considered negative and will not be compared as equal to a 'F' or 'C' sign.
	T"x"	Single text string that is case insensitive.
	T 'x(, xx,...)'	Multiple text strings, an 'OR' condition will be processed on each string.

data-type	EQP	Valid packed.
	NEP	Not packed.
	EQN	Valid external decimal numeric.
	NEN	Not external decimal numeric.
	Examples of EQP, EQN, NEP and NEN:	
	IF(10,5,EQP)	Will test for one packed field, five bytes (9 digits plus sign) in length, beginning in position10. If such a field is found, the record is selected.
	IF(100,0,5EQP)	Will test for five consecutive packed fields, 1 to 10 bytes (1 to 19 digits) in length. The 0 length is used to designate a field of any valid packed length. If the five fields are found, the record is selected.

	IF(150,5,EQN)
	
	IF(150,1,5EQN)
	
	IF(150,0,5EQN)
		Will all test for five consecutive numeric bytes (X'F0' through X'F9'). Note that if 0 is coded as the field length, it is ignored; field length will be considered 1 byte. If five numeric bytes are found, the record is selected.

	IF(150,5,5EQN)	Will test for 25 consecutive numeric bytes. If 25 numeric bytes are found beginning in position 150, the record is selected
	IF(15,3,NEP)	Will test for a 3 byte (5 digits plus a sign) packed field, beginning in position 15. If the data in these three bytes is not valid packed data, the record is selected.
	IF(135,6,NEN)	Will test for 6 characters beginning in position 135 for numeric data. If non-numeric data is found in any of the six positions, the record will be selected for processing.
segname	SEGMENT NAME	Specify name of segment to compare for in a valid data format. Most common format is C'xxxx'. Example: IF(SEG,EQ,C'PARTROOT')

IN

Used to limit the maximum numbers of segments or database records to be read from the input source.

This operand should only be specified once per command.

When used with IMS –mode processing signaled by use of one of the following commands:

ACCUMIMS, **COPYIMS**, **PRINTIMS**, **FPRINTIMS**, **LISTIMS**, **DUMPIMS**

the IN operand limits the number of database records read from **INPCB(..)**.

In non IMS-mode, the IN operand limits the number of segments or records that are read from the input source, either **INPCB(..)** or **INFILE(..)**.

Syntax:

IN(number)

Suboperands:

0	No limit.
number	A whole number

INFILE

Specify the input file name.

Use this operand when the input file is an unloaded copy of the database. Use this operand when the input file is SAM or VSAM.

This operand should only be specified once per command. This operand should not be coded in combination with the **INPCB** operand. Reserved ddnames as listed in the Allocations table on page 6 should not be used.

Syntax:

INFILE(ddname)

Sub-operands:

ddname	File name of input data set.

INPCB

Specify the input database PCB name contained in the PSB coded within the DFSRRC00 parameter list.

Use this operand when the primary input is a database.

This operand should only be specified once per command. This operand should not be coded in combination with the **INFILE** operand. If no input operand is coded, the default is INPCB(#1).

Syntax:

INPCB({pcbname|#pcbnumber})

Sub-operands:

pcbname	Database PCB name of primary input
#pcbnumber	The relative position number of the database PCB within the PSB. Precede the number with the number sign,#, Example #2. Non-database PCBs are not included when determining the relative position number.

INRID

INRID positions the **INPCB** or the **INFILE** file to a specific root segment.

The record identification field (**RID**) or root segment key is used for this positioning.

If the **INFILE** is not in IMS format mode, see chart below for **RID** value format.

This operand should only be specified once per command. This operand is only valid with the **COMPARE** command.

Syntax:

INRID(rid|root segment key)

Sub-operands:

rid	For ESDS, this is a 4 byte (`X'hhhhhhhh'`) hexadecimal RBA.
	For KSDS, this is a character, or hexadecimal, full or partial key.
	For RRDS, this is a numeric relative record number.
	For SAM, this is a numeric actual record number.
root segment key	Key of a root segment in the database or unloaded file.

JDATES

Calculate the number of days between the first and second dates.

This value can be either positive (if second date is greater) or negative (if second date is less). This value is then used in any subsequent **CALCDATE** operands where an asterisk is used in place of the **num_days** sub-operand.

This operand is only valid with the **CALCDATE** command.

Syntax:

JDATES(yyyy/ddd,yyyy/ddd)

Sub-operands:

yyyy/ddd	Beginning Julian date.
yyyy/ddd	Ending Julian date.

LIMIT

Used to limit the number of selected segments or database records that are written to the primary output of the command.

When used with IMS-mode processing, initiated by use of one of the following commands

COPYIMS, **PRINTIMS**, **LISTIMS**, **FPRINTIMS**, **DUMPIMS**

the limit is applied to the entire database record. In non IMS-mode, the limit is applied to the number of segments or records written to the output.

The **LIMIT** operand is independent of any **OUT** operands coded and supercedes it. Once the **LIMIT** count is reached, processing of the command terminates.

This operand may be coded one time per command.

Syntax:

LIMIT(number)

Sub-operands:

0	No limit.
number	A whole number, maximum number of records to write to output.

LIST

Print selected segments. Segments are printed in a horizontal character format.

To list records from an input file that is not in IMS format, use **FORM(NOIMS)**.

The segment is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT. See **COPY with LIST** example on page 99.

Syntax:

LIST(number)

Sub-operands:

0	No limit on number of segments printed.
number	Limit the number of segments printed.

MAPDD

MAPDD specifies the file name of the mapping criteria data set to be used for mapping the segments with the formatted print features (see also **FPRINT** and **COMPARE** commands and the **FPRINT(..)** operand).

The mapping criteria data set is created using MAX IMS/UTIL Batch. This allows for a unique copybook for each segment. When mapping criteria contains a segment layout definition which specifies a copybook member only without a library, the member is retrieved from the libraries allocated to the **COPYLIBS** ddname.

If a **MAPDD** is not specified when **FPRINT** and **COMPARE** are used, default criteria is used where each segment's copybook is assumed to be in a member with the same name as the segment and located in a data set defined by the **COPYLIBS** ddname.

COPYBOOK( ), **MAPDD( )**, **PLICOPY( )** are mutually exclusive, choose only one. This operand should only be specified once per command. See **FPRINT** example on page 102.

Syntax:

MAPDD(ddname)

Sub-operands:

ddname	File name of a data set and member that contains mapping criteria.

Reserved ddnames as listed in the Allocations table on page 6 should not be used.

MATCH

MATCH operand allows for selection of records in the input file (INFILE) to be based upon data found in a second file (**MTCHFILE**). Multiple **MATCH** operands may be used, however, if any one is found not to match, the record is not selected.

Note: For the **MATCH** process to work properly, both files must be sorted in ascending sequence based upon the fields used to do the match.

Boolean logic is allowed in combination with the **MATCH** logic, however, the Boolean logic will occur prior to the **MATCH** logic. Therefore, a record must first be selected by the Boolean (**IF, AND, OR**) logic prior to being processed by the **MATCH** logic.

Syntax:

MATCH(inloc, length, mtchloc)

Sub-operands:

inloc	Position in the INFILE to begin the **MATCH**.	
length	Length of the field to be **MATCH**ed.	
	`0`	**MATCH** rest of record using the MTCHFILE record length.
mtchloc	Position in the MTCHFILE containing the data to match against the INFILE.	

Example:

MATCH(1,9,25)

Would select records to be processed where the data in position `1` for a length of `9` of the **INFILE** was equal to the data in position `25` for a length of `9` in the **MTCHFILE**.

MATCHKEY

MATCHKEY operand allows for keyed selection of a database record in the input database (**INPCB**) to be based upon a root key value found in a second file (**MTCHFILE**).

The database record in the **INPCB** is read directly using the root key constructed from the data in the **MTCHFILE** as defined by this operand.

Use the **WARNMATCHKEY** operand to set the limit for the maximum number of missing database records on the **INPCB** or to eliminate the warning.

Syntax:

MATCHKEY(loc,len[,loc2,len2,...])

Sub-operands:

loc	Position in MTCHFILE containing root key data.
len	Length of data in MTCHFILE at this location.
loc2...	Position of additional location of data in MTCHFILE used to construct the root key.
len2...	Length of data at corresponding position.

Note: A single root key is constructed from the location and length values specified. The root key is constructed in the exact order as specified in the operand.

Example:

MATCHKEY(10,5,1,3)

Would construct an eight byte root key using data at location `10` for a length of `5` followed by data at location `1` for a length of `3` and use that to read a database record from the INPCB.

MATCHKEYRC

Set the return code value for any command using the MATCHKEY operand to select database records by root key value from the INPCB. If any database records are not found, this return code value will be used for that command. This value overrides the MAXIBAT default and any installation default for this step. See "Appendix D: Changing Installation Defaults" on page 119.

See the WARNMATCHKEY operand to eliminate counting the number of database records that are not found. If **WARNMATCHKEY** is set to `OFF`, this operand is ignored.

Syntax:

MATCHKEYRC(n)

Sub-operands:

n	Numeric value for return code.

Example:

MATCHKEYRC(8)

If database records to be selected using the MATCHKEY are not found on the INPCB for any of the following commands, the return code will be set to `8`.

MTCHFILE

Specify the match file name. This file may be a SAM file. This file contains the match criteria used to select records from the input file (INFILE) or input database (INPCB).

This operand is allowed once per command.

Syntax:

MTCHFILE(ddname)

Sub-operands:

ddname	File name of match data set.

Example:

MTCHFILE(MTCHIN)

Where **MTCHIN** is the ddname of the file containing the MATCH or MATCHKEY values.

MOVE

Builds an output record. The entire output record must be built with one or more move operands. The output record length will be determined by the last position moved to the output record. The RECFM and LRECL of the output file may cause the output record to be truncated or padded if necessary.

Output can be built in one of two ways. If the record is being built in IMS format, then the prefix is moved to the output record at initiation of its build. Therefore if a single output record is built from multiple input segments, it will carry the prefix of the first segment on the output record. To build output records that do not carry the segment prefix, use the **FORM(NOIMS)** operand.

Note: The ouput buffer is not cleared until the record is written, allowing the output record to be built from multiple input records.

Syntax:

MOVE(to-position,[dupl]to-data)
or, MOVE(to-position,length,from-position)

Sub-operands:

to-position	Position to store data.	
	`1-32760`	The actual position number.
	`+nnn or -nnn`	The relative position from the last selected position.
length	Length of data to store.	
	`0`	Rest of record.
dupl	Number of repeating occurrences of data (`C"...",X"...",P"..."`). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	
to-data	`C"x"`	Character.
	`X"hh"`	Hexadecimal.
	`P"nn"`	Packed, all leading zeroes should be specified.
from-position	Position of data in input record to copy.	
	`1-32760`	The actual position number.
	`+nnn or -nnn`	The relative position from the last selected position.

NOSELRC

Set the return code value for any command that follows in this same step where no records or segments are selected from the **INFILE** or **INPCB** using the selection criteria (**IF, AND, OR**). This value overrides the MAXIBAT default value or any installation default value. See "Appendix D: Changing Installation Defaults" on page 119.

Syntax:

NOSELRC(n)

Sub-operands:

n	Numeric value for return code.

Example:

NOSELRC(4)

If no records or segments are selected from the **INFILE** or **INPCB** for processing, the return code value is set to `4`.

OUT

Is used to set the maximum number of segments written to the output file or to SYSLIST.

OUT can be used to control the output count of individual segments when used in combination with the **IF** operand. **OUT** does not look at entire database records. Refer to the **LIMIT** operand on page 73.

Syntax:

OUT(number)

Sub-operands:

0	No limit.
number	A whole number.

OUTFILE

Specify the output file name. Use this operand to unload the database to a SAM file. VSAM output is also supported, however, the user must be aware of key values if KSDS file type is in use. This unloaded file can be used to reload the database.

When processing in IMS mode (**COPYIMS**) and segments are selected from the database, complete database records will be written in this file. This operand should only be specified once per command. It is ignored for any command other than the **COPY** command. This operand should not be coded in combination with the **COMPPCB**.

Syntax:

OUTFILE(ddname)

Sub-operands:

ddname	File name of output data set.

Reserved ddnames as listed in the Allocations table on page 6 should not be used.

OUTPATH

The **OUTPATH** operand is where you specify the UNIX file name for the output of data transformation.

Syntax:

```
OUTPATH(UNIXfilename)
```

Sub-operands:

UNIXfilename	is the entire pathname of the UNIX file that the output will be written to.

This operand is intended for use with the **COPY** command when the input data is transformed using a **FORMAT** option. Due to the size of a UNIX file name, this operand can continue over multiple control statements. Use a comma to show continuation.

Example:

```
OUTPATH(/u/mx10003/My_Outfile)
OUTPATH(/u/mx10003/filename_to_,
        be_very_long)
```

OUTPCB

Specify the output database PCB name contained in the PSB coded with the DFSRRC00 parameter list. Use this operand when the output of the copy command is to be a database. Input to this command can be another database (**INPCB**) or unloaded segments (**INFILE**) from a database. Unloaded segments must have been written with MAX IMS/UTIL Batch to ensure they contain the correct control information in order to reload them into the database.

This operand should only be specified once per command.

This operand should not be coded in combination with the **OUTFILE** operand. If the command being coded is not a form of **COPY**, this operand is ignored. If the input is from **INPCB()**, both PCBs must be within the same PSB.

Syntax:

OUTPCB({pcbname|#pcbnumber})

Sub-operands:

pcbname	Database PCB name of primary output.
#pcbnumber	The relative position number of the database PCB within the PSB. Precede the number with the number sign,#, Example #2. Non-database PCBs are not included when determining the relative position number.

PADCHAR

The pad character to be used to initialize any part of a segment or field that was left uninitialized from data manipulation operands. For example, a **CHANGE** operand that changes a 10 byte field to a 6 byte string. In this case, the segment or record would be extended to its full length with the insertion of the **PADCHAR**.

This operand should only be specified once per command.

Syntax:

PADCHAR({char|hex-char|X'00'})

Sub-operands:

char	Any single character.
hex-char	A single character represented in hex as follows: `X'hh'`.

PIN

PIN is used to specify a value or values to use in combination with the **SCRAMBLE** and **UNSCRAMBLE** operands to encode and decode data. Only one **PIN** operand may be coded. Its value applies to all **SCRAMBLE** and **UNSCRAMBLE** operands.

Syntax:

PIN(pin#{,pinpos,pinlen|KEY})

Sub-operands:

pin#	A numeric value used to build the encoding or decoding table. Maximum value: `999999`.
pinpos	Position in segment to extract a value to combine with the `pin#` to build encoding/decoding tables.
pinlen	Length of value in segment.
KEY	Use the segment key to build the encoding tables.

Note: The same values are required to decode (**UNSCRAMBLE**) data that were used to encode (**SCRAMBLE**) the data. If these values are not the same, the data cannot be decoded.

The `pin#` is required. If a segment value (`pinpos,pinlen`) is indicated, it will be used with the `pin#` to build the encoding/decoding tables. If `KEY` is coded, the key value will be used for this purpose. For IMS databases, this is the DBD defined segment key (if any); for KSDS files, this is the actual record key. For ESDS files, this is the RBA; and for RRDS files, this is the relative record number. Note that `KEY` is not valid for sequential files. If an IMS database is unloaded and the `KEY` is used to **SCRAMBLE**, you should use `KEY` to **UNSCRAMBLE** the unloaded database. If a KSDS file is unloaded to a sequential file and the `KEY` is used to **SCRAMBLE**, it is necessary to **UNSCRAMBLE** the file using `pinpos` and `pinlen`. This value will be shown in an informational message when the **SCRAMBLE** operation is performed.

Example:

PIN(50,KEY)

Will use the value of `50` along with the value of the key in each segment to **SCRAMBLE** or **UNSCRAMBLE** the data.

PLICOPY

PLICOPY identifies the data set and member name of the PL/I record layout to be used for mapping the segments with the formatted print features (see also **FPRINT** and **COMPARE** commands and the **FPRINT(..)** operand).

COPYBOOK(), **MAPDD**(), **PLICOPY()** are mutually exclusive, choose only one.

This operand should be specified only once per command.

Syntax:

PLICOPY(dsn(mem))

Sub-operands:

dsn(mem)	Fully qualified data set and member name to a PL/I layout.

PRINT

Print selected segments. Include the segment key and length in the report.

The report is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT.

To print records from an input file with records not in IMS format, use **FORM(NOIMS)**.

Syntax:

PRINT(number)

Sub-operands:

0	No limit on number of segments printed.
number	Limit the number of segments printed.

REFDD

REFDD specifies the file name of the reformatting criteria data set to be used for reformatting segments with the copy command. The reformatting criteria data set must have been created by MAX Data/Util (refer to the section 7. Build Reformat Criteria on page 96 of the MAX Data/Util User Reference Manual). When this operand is specified, the **COPYBOOK**, **PLICOPY**, or **MAPDD** must also be specified. See the **REFORMAT** example on page 106.

This operand should only be specified once per command. Reserved ddnames as listed in the Allocations table on page 6 should not be used.

Syntax:

REFDD(ddname)

Sub-operands:

ddname	File name of a data set and member which contains reformatting criteria.

REPLACE

Overlays data in a segment. The length of the segment does not change.

REPLALL is used to overlay *every* occurrence of the **from-data** found within the specified length of the record.

Syntax:

```
        REPLACE(position,[dupl]to-data)
or,     REPLACE(position,length,[dupl]from-data,[dupl]to-data)
or,     REPLACE(position,oper,[dupl]from-data,[dupl]to-data)
or,     REPLACE(position,length,[dupl]from-data,to-position,[dupl]to-data)
or,     REPLACE(position,oper,[dupl]from-data,to-position,[dupl]to-data)
or,     REPLALL(position,length,[dupl]from-data,[dupl]to-data)
```

Sub-operands:

position	Positions to begin scan.	
	`1-32760`	The actual position number.
	`+nnn` or `-nnn`	The relative position from the last selected position.
length	Length to scan.	
OR	`0`	Rest of segment.
operator	`EQ`	Equal.
	`NE`	Not equal.
	`GT`	Greater than.
	`GE`	Greater equal.
	`LE`	Less than or equal.
	`LT`	Less than.
dupl	Number of repeating occurrences of data (`C"..."`,`X"..."`,`P"..."`). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	
from-data	`C"x"`	Character case sensitive.
	`C'xx[,xx, ...]'`	Separate character strings with '`OR`' condition to be used for processing each string.
	`X"hh"`	Hexadecimal.
	`P"nn"`	Packed, all leading zeroes should be specified.
	`T"x"`	Case insensitive.
to-position	Position to overlay to-data.	
	`1-32760`	The actual position number.
	`+nnn` or `-nnn`	The relative position from the last selected position.
to-data	`C"x"`	Character.
	`X"hh"`	Hexadecimal.
	`P"nn"`	Packed, all leading zeroes should be specified.

RID

RID positions the input database or file to a specific segment. The record identification field (**RID**) or root segment key is used for this positioning.

If the input is not a database or an unloaded database being processed, see chart below for rid format.

This operand should only be specified once per command.

Alias: RBA RRN KEY

Syntax:

RID(rid | root segment key)

Sub-operands:

rid	For ESDS, this is a 4 byte (`X'hhhhhhhh'`) hexadecimal RBA.
	For KSDS, this is a character, or hexadecimal, full or partial key.
	For RRDS, this is a numeric relative record number.
	For SAM, this is a numeric actual record number.
root segment key	Full or partial key of root segment in the database or an unloaded file (IMS mode). HDAM requires a full key value.

SCRAMBLE

Encodes the data in the designated positions of the segment using an algorithm to maximize data privacy. Data encoded using this operand can later be decoded using the **UNSCRAMBLE** operand.

Syntax:

SCRAMBLE(loc,len{,C'string'|X'string'|reserved word})

Sub-operands:

loc	Location of data to encode.
len	Length of data to encode. `0` = length to end of segment.
c'string'	Supply list of characters to be encoded.
x'string'	Supply list of hex characters to be encoded.
reserved word	Specific reserved words indicating type of characters to be encoded.

When neither a data string nor a reserved word is specified, all characters will be involved in the encoding.

Reserved Words:

ALPHA	Includes upper and lowercase letters.
NUMERIC	Includes all digits, 0-9.
ALPHAMERIC	Includes upper and lowercase letters and all digits, 0-9.
UPPER	Includes uppercase letters only.
LOWER	Includes lowercase letters only.
PACKED	Includes valid packed numeric values, sign will not change.

A pin value is required to encode the characters. See the **PIN** operand on page 82.

Examples:

```
SCRAMBLE(1,0)
```

This would encode the entire segment because the character set is not limited to a reserved word or string. All characters would be encoded.

```
SCRAMBLE(15,5,NUMERIC)
```

This would encode all numeric characters starting at location 15 for 5 in length.

SELECT

Selects every nth input segment or database record (IMS mode).

When processing in IMS-mode (commands **ACCUMIMS**, **COPYIMS**, **PRINTIMS**, **COPYIMS**, **FPRINTIMS**, **DUMPIMS**), the frequency of records chosen by the **SELECT** operand is determined prior to the application of any specific selection criteria, **IF, AND, OR** operands.

Syntax:

```
SELECT(number)
```

Sub-operands:

number	A whole number.
0	No selection in effect.

SETMAXRC

Set the value of the maximum return code at this point in the command set. A return code is set for each command. When multiple commands are contained in the same step, the maximum return code value is kept and used as the return code for the step. This operand forces the value of this maximum return code without regard for the prior value. Subsequent commands could cause a change in that value. To force a return code value for the entire step, code the **OPTION** command as the last command in the step using the **SETMAXRC** at that time.

Syntax:

SETMAXRC(n)

Sub-operands:

n	Numeric value for return code.

Example:

SETMAXRC(0)

Will force the maximum return code value to zero.

SETRC

Set a specific return code at the end of processing the command.

When used with the **OPTION** statement, will not only set the value of the return code at that point in the set of commands, but will be tested to determine if it is greater than the current maximum return code. If this is found to be true, the value of the maximum return code will also be set to this value.

Syntax:

SETRC(number)

Sub-operands:

number	A whole number that will be used as the return code at end of processing.

Example:

SETRC(0)

Will force the return code to be set to `0` upon completion. Reports will reflect the actual condition code, but it will be set to this value upon return.

SKIP

Bypass given number of segments.

Note: This operand is valid only with the **ALL** options of the commands.

Syntax:

SKIP(number)

Sub-operands:

number	A whole number indicates total # of segments to bypass.
0	Indicates no limit.

SORTPARM

Pass actual parameters to the **SORT** program. The parameters passed should reflect the segment after it has been copied and manipulated by the given operands.

For example, if segments in the input file or database are to be sorted by the data in columns `20` through `25`, but a **CHANGE** operand expanded a field prior to column `20` by `2` bytes, then the actual columns that would be stated in the **SORTPARM** would be `22` through `27`.

Syntax:

SORTPARM(actual sort parameters)

Example:

SORTPARM(SORT FIELDS(22,6,CH,A))

Would sort the output file into ascending sequence based on the character data in columns `22` through `27`.

Note: It is important to remember that any parameters coded for the **SORT** within the **SORTPARM** operand are not changed by MAXIBAT. They are passed to the **SORT** for validation. The **FIELDS** parameter must be coded to the **SORT** specifications. The first data byte of a fixed file is `1`, the first data byte of a variable file is `5`. The record descriptor word must be accounted for in the variable file.

STOP

Specifies selection criteria to be used to establish a position in the input to stop processing. This operand should only be specified once for a command.

When multiple selection criteria is specified within the **STOP** operand they are **OR**'ed together. When any single criteria passes the test against the data record the entire **STOP** is considered successful.

Syntax:

```
        STOP(position,length,[dupl]data[,pos,len,data, . . .])
or,     STOP(position,operator,[dupl]data[, . . .])
or,     STOP(position,length,data-type[, . . .])
or,     STOP(SEG,operator, segname)
```

Sub-operands:

position	Position in segment.	
	`1-32760`	The actual position number.
	`+nnn` or `–nnn`	The relative position from the last selected position.
SEG	Indicates to compare for specific segment name.	
length	Length to scan, use of length implies equal condition for compare.	
	`0`	Rest of record.
operator	`EQ`	Equal.
	`NE`	Not equal.
	`GT`	Greater than.
	`GE`	Greater equal.
	`LE`	Less than or equal.
	`LT`	Less than.
dupl	Number of repeating occurrences of data (`C"..."`, `X"..."`, `P"..."`). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	

data	`C"x"`	Single character string that is case sensitive.
	`C'xx[,xx, ...]'`	Multiple character strings, an 'OR' condition will be used for processing each string.
	`X"hh"`	Single hexadecimal string.
	`X'xx[,xx, ...]'`	Multiple hexadecimal strings, an 'OR' condition will be used for processing each string.
	`P"nn"`	Single Packed decimal string.
	`P'xx[,xx, ...]'`	Multiple Packed decimal strings, an 'OR' condition will be used for processing each string.
	`T"x"`	Single Text string that is case insensitive.
data-type	`EQP`	Valid packed.
	`NEP`	Not packed.
	`EQN`	Valid external decimal numeric.
	`NEN`	Not external decimal numeric.
segname	`SEGMENT-NAME`	Specify name of segment to compare for in a valid data format. Most common format is `C"xxxx"`. Example: `IF(SEG,EQ, C'PARTROOT')`

STOPMAXRC

Will set a condition to be tested at the completion of every command that follows to determine if the remaining commands in the step should be bypassed. This condition is tested using the maximum return code value.

Syntax:

STOPMAXRC(cond,value)

Sub-operands:

cond	**EQ**, **NE**, **LT**, **LE**, **GT**, **GE**.
value	Numeric value to test.

Examples:

STOPMAXRC(NE,0)

If the maximum return code is found to not equal zero, the remainder of the commands will be bypassed.

STOPMAXRC(GT,4)

If the maximum return code is found to be greater than 4, the remainder of the commands will be bypassed.

STOPRC

Will set a condition to be tested at the completion of every command that follows to determine if the remaining commands in the step should be bypassed. This condition is tested using the current command's return code value. If **STOPRC** is coded on an **OPTION** statement that is the first in the command sequence, it has the same effect as **STOPMAXRC**. However, if the **OPTION** statement containing this operand follows other commands, the test will not be impacted by any prior condition codes and will cause the remaining commands to be bypassed once this condition is found to be true.

Syntax:

STOPRC(cond,value)

Sub-operands:

cond	**EQ**, **NE**, **GT**, **GE**, **LT**, **LE**.
value	Numeric value to test.

Example:

STOPRC(GE,8)

If the return code from any of the commands that follow is found to be greater than or equal to 8, the remaining commands will be bypassed.

SUBSYS

Specify subsystem name and optional parameter data for processing copybooks. If your copybooks reside in a repository that requires a subsystem to allow retrieval as opposed to a standard PDS, use this operand to specify the name.

Syntax:

SUBSYS(name)
SUBSYS(name, parameter)

Sub-operands:

name	Subsystem name.
parameter	Optional parameters to pass to subsystem.

TRANSLATE

Will translate a set of characters in the segment to a new set of characters.

Syntax:

TRANSLATE(loc,len,{external name|target string}{,select string}{,pad})

Sub-operands:

loc	Location of data to translate.
len	Length of data; `0` = to end of segment.
external name	Name of translate table in MAXDFLTS; see Appendix D: Changing Installation Defaults.
target string	Character string representing the character to translate the corresponding character in the select string.
select string	Character string representing all characters to be translated.
pad	Pad character.

Characters within the loc and length are translated. If the optional select string is coded, only characters matching a character in that string will be selected for translation. Characters are translated to the corresponding character in the target list or external table name specified. If a target string is specified, a corresponding selection string must also be specified. If an external table name is coded with no selection string, all characters will be translated as indicated by the external table. If the target string contains fewer characters than the selection string, all remaining characters will be translated to the pad character. The pad character can be coded as part of the translate or see the **PADCHAR** operand.

Examples:

TRANSLATE(1,0,TRT#ALPH)

Will translate all characters in the segment to the values coded in the external translate table in MAXDFLTS named TRT#ALPH. An error will occur if this table cannot be found.

TRANSLATE(10,20,C'JKLMN', C'ABCDE')

Will translate characters in the selection string to the corresponding characters in the target string. In this case, `A` will translate to `J`, `B` to `K`, `C` to `L`, `D` to `M` and `E` to `N`.

See MAXDFLTS in "Appendix D: Changing Installation Defaults" on page 119 for details on coding custom translate tables.

TRANSRC

Set the return code value for any command where data cannot be transformed due to missing copybooks. This return code is only valid if the **WARNTRANERR** operand is set to numeric value to track the transformation errors. If the **WARNTRANERR** operand is set to `OFF`, no tracking will occur therefore this return code has no meaning.

This value overrides both the MAXIBAT default and any installation default that has been set in the MAXDFLTS. See "Appendix D: Changing Installation Defaults" on page 119.

Syntax:

TRANSRC(n)

Sub-operands:

n	Numeric value for return code.

Example:

TRANSRC(8)

If segments cannot be translated due to missing copybooks, the return code will be set to `8`.

TRUNCRC

Set the return code value for any command where the output segments (one or more) are truncated. This value overrides both the MAXIBAT default and any installation default set using the MAXDFLTS. See "Appendix D: Changing Installation Defaults" on page 119.

Syntax:

TRUNCRC(n)

Sub-operands:

n	Numeric value for return code.

Example:

TRUNCRC(0)

If segment truncation occurs, return zero as the return code, do not cause any warnings or errors to be activated. The number of truncated segments will still be provided.

UNSCRAMBLE

Decodes the data in the designated positions of the segment that was encoded using the **SCRAMBLE** operand.

Syntax:

UNSCRAMBLE(loc,len{,C'string'|X'string'|reserved word})

Sub-operands:

Loc	Location of data to decode.
Len	Length of data to decode. 0 = length to end of segment.
c'string'	Supply list of characters to be decoded.
x'string'	Supply list of hex characters to be decoded.
reserved word	Specific reserved words indicating characters to be decoded.

When neither a data string nor a reserved word is specified, all characters will be involved in the decoding.

Reserved Words:

ALPHA	Includes upper and lowercase letters.
NUMERIC	Includes all digits, 0-9.
ALPHAMERIC	Includes upper and lowercase letters and all digits, 0-9.
UPPER	Includes uppercase letters only.
LOWER	Includes lowercase letters only.
PACKED	Includes valid packed numeric values, sign will not change.

A pin value is required to decode the characters. See the **PIN** operand on page 82.

Examples:

UNSCRAMBLE(1,0)

This would decode the entire segment because the character set is not limited to a reserved word or string. All characters would be decoded.

UNSCRAMBLE(15,5,NUMERIC)

This would decode all `numeric` characters starting at location `15` for `5` in length.

Note: The **PIN** operand used with any **UNSCRAMBLE** must be the same as that used in the **SCRAMBLE**.

WARNMATCHKEY

Is used to turn the **MATCHKEY** error warning `ON` or `OFF`, and to supply a limit to the number of mismatches before terminating the current command. If the **WARNMATCHKEY** is set to `OFF`, no warnings will be given or accumulated and the **MATCHKEYRC** will be ignored. If the **WARNMATCHKEY** is set to a numeric value, the **MATCHKEYRC** value will be used if any mismatches are found. A value of zero indicates that all mismatches should be noted but the command is to continue. A numeric value other than zero will both warn of mismatches and terminate the command once the given number of mismatches has occurred.

A mismatch condition occurs when the value of the root key constructed by the **MATCHKEY** operand does not match the root key of the database record in the **INPCB**. If the constructed key is generic, this indicates no match on the generic value.

The setting of this operand overrides both the default value of MAXIBAT and any installation default that has been set by MAXDFLTS. See "Appendix D: Changing Installation Defaults" on page 119.

Syntax:

WARNMATCHKEY(OFF|value)

Sub-operands:

OFF	Turn off warning of mismatches.
value	Numeric value limit of mismatches prior to termination. 0 = no limit on mismatches.

Example:

WARNMATCHKEY(30)

Will continue processing if mismatches occur, up to a maximum of 30. Once 30 mismatches occur, the command will terminate. The return code for the command will be set using either the MAXIBAT default, the installation default, or the value specified by **MATCHKEYRC** should any mismatches occur.

WARNTRANERR

Is used to turn the error warning on data transformation errors ON and OFF, and to set a limit as to the number of errors encountered before the command is terminated. If **WARNTRANERR** is set to OFF, transformation errors caused by missing copybooks are ignored. If a numeric limit is set, the command will terminate once the limit has been reached. If the limit is set to zero, the number of errors will be accumulated but will not cause early termination. If transformation errors are being counted and do occur, the return code will be set by either the MAXIBAT default, the installation default or the **TRANSRC** value. The setting of this operand overrides both the default value of MAXIBAT and any installation default that has been set by MAXDFLTS. See "Appendix D: Changing Installation Defaults" on page 119.

Syntax:

WARNTRANERR(OFF|value)

Sub-operands:

OFF	Turn off warning of transformation errors.
value	Numeric limit of transformation errors prior to early termination of the command. 0 = no limit on errors, report on total.

Example:

WARNTRANERR(0)

Process the entire command, count the number of transformation errors. The 0 value sets the number of errors accepted to unlimited. The return code for the command will be set from the MAXIBAT default, the installation default, or the **TRANSRC** operand if errors do occur in transformation.

WRITE

Directs selected record to be written to a specific output file name. The output file(s) identified by this operand may be either VSAM or SAM.

This operand may be used to write selected segments to one file while writing other selected segments to another during a single pass of the input database or file. Reserved ddnames as listed in the Allocations table on page 6 should not be used.

Syntax:

```
WRITE(ddname1[,ddname2] . . .)
```

Sub-operands:

ddname1	File name of output data set.
ddname2	File name of output data set.

Example:

```
ACCUM INPCB(#1),
          IF(SEG,EQ,C"PARTROOT')  WRITE(SYSUT2),
          IF(SEG,EQ,C'BACKORDR')  WRITE(SYSUT3)
```

Write all root segments to SYSUT2 . Write all back order segments to SYSUT3.

Note: IMS mode is not supported by the **WRITE** operand. Only segments that match the criteria for selection are written to this file. The file created by the use of the **WRITE** operand may not be subsequently reloaded.

CHAPTER 5: EXAMPLES

COPY with LIST

The following example shows output of a copy in two formats.

The first **COPY** will unload database segments. The **OUTFILE** will be in IMS format. Records written in this format contain a control prefix.

```
COPY INPCB(#1) OUTFILE(SYSUT2)
```

Note prefix information:

```
PARTROOT01    02AN960C00          DRYER14
PARTROOT01    02AN960C13          DRYER12
STANINFO02    02          742                         1201  15      A6C      06C
STOKSTAT02    00 AA16511    000000000     EACH  0000000110000000                  512    0000000{0000131{{0000000{0 Z512N
STOKSTAT02    00 AA16512    000000000     EACH  0000000110000000                  512    0000000{0000131{0000000{0 Z512N
STOKSTAT02    00 AK2877F    M000100000    EACH00000000000002270000                  360    0000000{0000088{{0000000{0 Z360N
STOKSTAT02    0028009126    000000000     EACH            000000000000000000000051351751750000000000000630{0000000000000000{0  494Y
PARTROOT01    02AN960C14          DRYER12
STANINFO02    02          742                         1201  15      A6C      06C
STOKSTAT02    00 AA16511    000000000     EACH  0000000110000000                  512    0000000{0000131{0000000{0 Z512N
STOKSTAT02    00 AA16512    000000000     EACH  0000000110000000                  512    0000000{0000131{0000000{0 Z512N
STOKSTAT02    00 AK2877F    M000100000    EACH00000000000002270000                  360    0000000{0000088{0000000{0 Z360N
STOKSTAT02    0028009126    000000000     EACH            000000000000000000000051351751750000000000000630{0000000000000000{0  494Y
PARTROOT01    02AN960C17          DRYERRERRRRRR
STANINFO02    02          742                         1201  15      A6C      06C
STOKSTAT02    00 AA16511    000000000     EACH  0000000110000000                  512    0000000{0000131{0000000{0 Z512N
STOKSTAT02    00 AA16512    000000000     EACH  0000000110000000                  512    0000000{0000131{0000000{0 Z512N
STOKSTAT02    00 AK2877F    M000100000    EACH00000000000002270000                  360    0000000{0000088{0000000{0 Z360N
STOKSTAT02    0028009126    000000000     EACH            000000000000000000000051351751750000000000000630{0000000000000000{0  494Y
PARTROOT01    02AN960C97          DRYERRERRRRRR6
```

Figure 10: Example of unloaded data

The second **COPY** will write segments without a prefix, **FORM(NOIMS)**. Only the output from the first copy can be used to reload a database.

```
COPY INPCB(#1) OUTFILE(SYSUT2) FORM(NOIMS)
```

```
02AN960C00              DRYER14
02AN960C13              DRYER12
02              742                         1201  15      A6C      06C
00 AA16511        000000000     EACH  0000000110000000                  512    0000000{{0000000{0 Z512N
00 AA16512        000000000     EACH  0000000110000000                  512    0000000{0000000{0 Z512N
00 AK2877F        M000100000    EACH000000000000270000                  360    0000000{0000000{0 Z360N
0028009126        000000000     EACH            00000000000000000000051351751750000000000000630{{0000000000000000{0
494Y
02AN960C14              DRYER12
02              742                         1201  15      A6C      06C
00 AA16511        000000000     EACH  0000000110000000                  512    0000000{0000000{0 Z512N
00 AA16512        000000000     EACH  0000000110000000                  512    0000000{0000000{0 Z512N
00 AK2877F        M000100000    EACH000000000000270000                  360    0000000{0000000{0 Z360N
0028009126        000000000     EACH            00000000000000000000051351751750000000000000630{000000000000000{0  494Y
02AN960C17              DRYERRERRRRRR
02              742                         1201  15      A6C      06C
00 AA16511        000000000     EACH  0000000110000000                  512    0000000{0000000{0 Z512N
00 AA16512        000000000     EACH  0000000110000000                  512    0000000{0000000{0 Z512N
00 AK2877F        M000100000    EACH000000000000270000                  360    0000000{0000000{0 Z360N
0028009126        000000000     EACH            00000000000000000000051351751750000000000000630{000000000000000{0  494Y
02AN960C97              DRYERRERRRRRR6
```

Figure 11: Example of data copied with **FORM(NOIMS)**

COPY with CHANGE

The following control statements were supplied to unload an entire database. Segments with the name 'PARTROOT' and the value of 'CAPACITOR' at any position within the segment are changed.

```
COPYALL INPCB(#1) OUTFILE(SYSUT2),
        IF(SEG,EQ,C'PARTROOT') AND(1,0,C'CAPACITOR  '),
        CHANGE(1,0,C'CAPACITOR  ',C'CAPACITOR-X'),
        PRINT(10)
```

```
MAX PRINT SERVICES                              DATE=15 Sep 1999 TIME=13:33:21                          PAGE 1

SEGMENT NAME = PARTROOT       KEY LENGTH=17 KEY POSITION=1
OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
     0 F0F2C3D2 F0F5C3E6 F1F8F1D2 40404040 40                                    *02CK05CW181K                    *
SEGMENT - LENGTH=50
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0----+---11----+---12
02CK05CW181K              CAPACITOR-X

SEGMENT NAME = PARTROOT       KEY LENGTH=17 KEY POSITION=1
OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
     0 F0F2F2F5 F0F2F3F6 60F0F0F1 40404040 40                                    *02250236-001                    *
SEGMENT - LENGTH=50
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0----+---11----+---12
02250236-001              CAPACITOR-X

SEGMENT NAME = PARTROOT       KEY LENGTH=17 KEY POSITION=1
OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
     0 F0F2F3F0 F0F9F2F2 F8404040 40404040 40                                    *023009228                       *
SEGMENT - LENGTH=50
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0----+---11----+---12
023009228                 CAPACITOR-X
```

Figure 12: **COPY** with **CHANGE** example

Note: By including the two spaces in the **CHANGE** operand compare value, the segment length is *not* affected when the new value is put into the segment. The first 10 segments selected to be changed are printed.

FPRINT

The following control statements were supplied to print selected database records from the input PCB using mapping criteria.

```
FPRINTIMS INPCB(#1) OUTFILE(SYSUT2),
          MAPDD(IMSMAP),
          IF(SEG,EQ,C'STOKSTAT') AND(4,EQ,C'AK2877F')
```

In this example the **OUTFILE** operand is ignored. This allows for an easy switch to a **COPYIMS** command following a review of the formatted print.

```
MAX PRINT SERVICES                              DATE=16 Sep 1999 TIME=08:48:05                          PAGE 1

SEGMENT NAME = PARTROOT       KEY LENGTH=17 KEY POSITION=1
OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
     0 F0F2C1D5 F9F6F0C3 F1F34040 40404040 40                                  *02AN960C13                     *
SEGMENT - LENGTH=50                 COPYBOOK DESC=PART DEFINITION
POSS  *----------FIELD NAME---------* FORMAT ---+----1----+----2----+----3----+----4----+----5----+----6----+----7
00001 DIV                           C    2 02
00003 ITEM-NO                       C   15 AN960C13
00027 ITEM-DESCR                    C   20 DRYER12

SEGMENT NAME = STANINFO       KEY LENGTH=2 KEY POSITION=1
OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
     0 F0F2                                                                    *02                             *
SEGMENT - LENGTH=85                 COPYBOOK DESC=STANDARD INFO_<=25
POSS  *----------FIELD NAME---------* FORMAT *---+----1----+----2----+----3----+----4----+----5----+----6----+----7*
00001 DIV                           C    2 02
00019 STAN-PROC-CODE                C    2 74
00021 INVENTORY-CODE                C    1 2
00022 PLANNING-REVISION-NUMBER      C    2
00048 MAKE-DEPT                     C    2 12
00050 MAKE-COST-CTR                 C    2 01
00054 COMMODITY-CODE                C    4 15
00062 MAKE-SPAN                     Z    3 X'C1F6C3'

SEGMENT NAME = STOKSTAT       KEY LENGTH=16 KEY POSITION=1
OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
     0 F0F040C1 D2F2F8F7 F7C64040 40404040                                     *00 AK2877F                     *
SEGMENT - LENGTH=160                COPYBOOK DESC=STOCK STATUS
POSS  *----------FIELD NAME---------* FORMAT *---+----1----+----2----+----3----+----4----+----5----+----6----+----7-*
00001 REG                           C    2 00
00003 LOCATION                      C    8  AK2877F
00021 QTY-ON-HAND                   Z  6.3 000100.000
00035 UNIT                          C    4 EACH
00051 ROP                           Z    3 270
00054 PLANNED                       Z    3 000
00057 MRP                           C    1
00058 /FILLER-5/                    C   32                  360    0000000{
00090 CURRNT                        Z  7.1 +0000088.0
00098 UNPLANNED                     Z  7.1 +0000000.0
00106 ON-ORDER                      Z  7.1 +0000000.0
00114 TOTAL-STOCK                   Z  7.1 0000088.0
00130 UNPLAN                        Z  7.1 0000000.0
00138 SPARES                        Z  7.1 +0000000.0
00146 DIVERS                        Z  7.1 +0000000.0
00154 /FILLER-7/                    C    7 0 Z360N

SEGMENT NAME = PARTROOT       KEY LENGTH=17 KEY POSITION=1
OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
     0 F0F2C1D5 F9F6F0C3 F1F74040 40404040 40                                  *02AN960C17                     *
SEGMENT - LENGTH=50                 COPYBOOK DESC=PART DEFINITION
POSS  *----------FIELD NAME---------* FORMAT *---+----1----+----2----+----3----+----4----+----5----+----6----+----7*
00001 DIV                           C    2 02
00003 ITEM-NO                       C   15 AN960C17
00027 ITEM-DESCR                    C   20 DRYERRERRRRRR
```

Figure 13: PRINT Example

Copybook members referenced without a library in the IMSMAP mapping criteria will be retrieved from the libraries allocated to the COPYLIBS ddname.

COMPARE

The following control statements were used to unload all segments (**COPYALL**) from the database. During the unload, a change is made to selected segments. The **COMPARE** command is then used to compare the original database to the unloaded file. Use of the **MAPDD** operand causes the output to be presented in formatted mode.

```
COPYALL INPCB(#1) OUTFILE(SYSUT2),
        IF(SEG,EQ,C'PARTROOT') AND(1,0,C'DRYERR'),
        CHANGE(1,0,C'DRYERR',C'WASHER')
        COMPARE INPCB(#1) COMPFILE(SYSUT2),MAPDD(IMSMAP)
```

```
MAX PRINT SERVICES                                DATE=14 Jan 2000 TIME=15:17:33                          PAGE 1

 SEGMENT MISMATCHES POS=43PART DEFINITION     INPCB(#1)      SEG#=8 LEN=66               COMPFILE(SYSUT2) REC#=8 LEN=66
 POSS  *----------FIELD NAME---------* FORMAT *---+----1----+----2----+----3----+----4-*  *---+----1----+----+----2----+----3
 00001 ROOT-FORMAT
 00001 DIV                            C    2 02
 00003 ITEM-NO                        C   15 AN960C17
 00027 ITEM-DESCR                     C   20 DRYERR                                      WASHER

 SEGMENT MISMATCHES POS=43PART DEFINITION     INPCB(#1)      SEG#=14 LEN=66              COMPFILE(SYSUT2) REC#=14 LEN=66
 POSS  *----------FIELD NAME---------* FORMAT *---+----1----+----2----+----3----+----4-*  *---+----1----+----+----2----+----3*
 00001 ROOT-FORMAT
 00001 DIV                            C    2 02
 00003 ITEM-NO                        C   15 AN960C97
 00027 ITEM-DESCR                     C   20 DRYERR6                                     WASHER6

 SEGMENT MISMATCHES POS=43PART DEFINITION     INPCB(#1)      SEG#=15 LEN=66              COMPFILE(SYSUT2) REC#=15 LEN=66
 POSS  *----------FIELD NAME---------* FORMAT *---+----1----+----2----+----3----+----4-*  *---+----1----+----+----2----+----3*
 00001 ROOT-FORMAT
 00001 DIV                            C    2 02
 00003 ITEM-NO                        C   15 AN960C98
 00027 ITEM-DESCR                     C   20 DRYERR6                                     WASHER6
```

Figure 14: **COMPARE** example

Copybook members referenced without a library in the IMSMAP mapping criteria will be retrieved from the libraries allocated to the **COPYLIBS** ddname.

MTCHFILE/MATCHKEY

This example shows use of the **MTCHFILE** and **MATCHKEY** operands. In the first **COPY**, MXROOT segments whose key matches the key provided in **MTCHFILE** will be copied. In the second COPYIMS, the root and all its dependent segments will be copied, based on the **MATCHKEY**. In the **FPRINT** command, I want a copybook formatted print of the matched segment.

```
//SYSIN    DD  *
 OPTION MATCHKEYRC(2)
    COPY INPCB(#1) OUTFILE(SYSUT2) MTCHFILE(MATCH1),
       MATCHKEY(1,7) IF(SEG,EQ,C'MXROOT');
    COPYIMS INPCB(#1) OUTFILE(SYSUT2) MTCHFILE(MATCH2),
       MATCHKEY(1,7) IF(SEG,EQ,C'MXROOT');
    FPRINT INPCB(#1) MTCHFILE(MATCH3),
       MATCHKEY(1,7) IF(SEG,EQ,C'MXROOT');
//
```

```
COPY Result :
  SEGMENTS READ: INPCB(#1)=6    COPIED: OUTFILE(SYSUT2)=6

COPYIMS Result :
  ROOTS SELECTED: INPCB(#1)=6    COPIED: OUTFILE(SYSUT2)=36

FPRINT Result :
SEGMENT NAME = MXROOT          KEY LENGTH=7 KEY POSITION=1
OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *---*
     0 F0F0F0F0 F0F5F5                                                    *
SEGMENT - LENGTH=102                 COPYBOOK DESC=MXROOT   SEGMENT
POSS  *----------FIELD NAME---------* FORMAT *---+----1----+----2----+----3
00001 MXS-ROOT-KEY                   N    7 0000055
00008 MXS-NAME                       C    7 MXROOT
00015 MXS-ROOT-DESC                  C   35 number 298 Root Level 1 segment
00050 MXS-ODD-EVEN                   C    4 ODD
00054 /FILLER-1/                     C    4
```

Figure 15: **MATCH** example

The `OPTION MATCHKEYRC(2)` statement indicates that when a key in the **MTCHFILE** is not located, the error message "`MAX236W MATCHKEY value xxxxxxxx not found`" will set a return code value of '`2`'. The default is RC is '`4`', unless changed via MAXDFLTS.

REFORMAT

This example shows 3 commands being executed within the same step.

```
===>                                                    SCROLL ===> CS
 //STEP1   EXEC PGM=MAXRRC00,REGION=4M,
 //           PARM=(BMP,MAXIBAT,MAXIBRED,,,C00000,,,,,,,,IVP3,
 //           IVP,,,,'XXXX')
 //STEPLIB   DD DISP=SHR,DSN=MXS.MXRXV310.LOADLIB
 //IMS       DD DSN=IMS.DBDLIB,DISP=SHR
 //           DD DSN=IMS.PSBLIB,DISP=SHR
 //SYSTSPRT  DD SYSOUT=*
 //SYSPRINT  DD SYSOUT=*
 //SYSLIST   DD SYSOUT=*
 //IMSMAP    DD DSN=MXS.IMS.COPYLIB(IMSMAP),DISP=SHR
 //IMSREF    DD DSN=MXS.TEST.MAAPDS(CYCLEREF),DISP=SHR
 //COPYLIBS  DD DSN=MXS.IMS.COPYLIBS,DISP=SHR
 //SYSUT2    DD DSN=&&TEMP,DISP=(,PASS),UNIT=SYSDA,
 //  SPACE=(TRK,(10,2)),DCB=(RECFM=VB,LRECL=180,BLKSIZE=6004)
 //SYSIN DD  *
 *  copy database and reformat the cycle count segments
   COPY INPCB(#1) OUTFILE(SYSUT2),
      REFDD(IMSREF) MAPDD(IMSMAP)
 *  print first cycle count segment from database
   FPRINT INPCB(#1) IF(SEG,EQ,C'CYCCOUNT'),
       COPYBOOK(MXS.TEST.MAAPDS(CYCCOUNT)) OUT(1)
 *  print first reformatted cycle count segment from unload file
   FPRINT INFILE(SYSUT2) IF(SEG,EQ,C'CYCCOUNT'),
       COPYBOOK(MXS.TEST.MAAPDS(NEWCOUNT)) OUT(1)
 //
```

Figure 16: REFORMAT Example

Note: MAX Data/Util is required to build reformat criteria.

COPY	This command unloads #1 to SYSUT2. Reformat criteria is used to re-arrange fields using COBOL copybooks to define segments to change
FPRINT	This command prints 1 record from the #1 PCB in formatted mode using a copybook to map the data
FPRINT	This command prints 1 record from the SYSUT2 file in formatted mode using a copybook to map the data

REFORMAT Output

```
MAX PRINT SERVICES                                DATE=17 Sep 1999 TIME=08:22:10                          PAGE 1
SEGMENT NAME = CYCCOUNT       KEY LENGTH=2 KEY POSITION=1
OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
     0 F2F0                                                                    *20                            *
SEGMENT - LENGTH=25                  COPYBOOK DESC=
POSS  *----------FIELD NAME----------* FORMAT *---+----1----+----2----+----3----+----4----+----5----+----6----+----7*
00001 /FILLER-1/                     C    2 20
00003 PHYSICAL-COUNT                 Z  7.1 0000360.0
00011 /FILLER-2/                     C    4
00015 TOTAL-STOCK                    Z  7.1 1234567.8
00023 /FILLER-3/                     C    3
  SEGMENTS READ: INPCB(#1)=61    SEGMENTS PRINTED=1
RECORD NUMBER=61
```

Figure 17: **REFORMAT** example: Original record before reformat

```
MAX PRINT SERVICES                                DATE=17 Sep 1999 TIME=08:22:10                          PAGE 1
SEGMENT NAME = CYCCOUNT       KEY LENGTH=2 KEY POSITION=1
OFFSET 0------* *------* 8------* *------* 16-----* *------* 24-----* *------*  0----+----1----+----2----+----3-*
     0 F2F0                                                                    *20                            *
SEGMENT - LENGTH=25                  COPYBOOK DESC=
POSS  *----------FIELD NAME----------* FORMAT *---+----1----+----2----+----3----+----4----+----5----+----6----+----7*
00001 /FILLER-1/                     C    2 20
00003 TOTAL-STOCK                    Z  7.1 +1234567.8
00011 /FILLER-2/                     C    4
00015 PHYSICAL-COUNT                 Z  7.1 +0000360.0
00023 /FILLER-3/                     C    3
  RECORDS READ: INFILE(SYSUT2)=61    PRINTED=1
_
```

Figure 18: **REFORMAT** example: New record after reformat

Note: The physical count and total stock fields have been rearranged.

Encoding/Decoding Data

The following is an example of copying database segments to a backup sequential file using **SCRAMBLE** to encode the data.

Prior to the **COPY**, sample list of segments:

```
H011111BETTY     TOE            J555 PLAIN DR.       Denver     CO....1997/03/24..
H011121Bettyjo   BialoskiII     H123 Vaughn Way      XYZ
H011122JOHN      TOE            H555 PLAIN DR.       DENVER     CO....1985/11/14..
H011133NANCY     JONES          F751 OAK STREET      DENVER     CO...%1961/03/12..
H011144JOHN      JONES          S751 OAK STREET      DENVER     CO....2002/07/22..
H011155MATHEW    SMITH          J607 JEFFERSON STREETHARVARD    IL....1997/07/17..
H011166SALLY     JOHNSON        K352 SUPERIOR AVE    SHAKER HT  OH. ..1965/01/31..
H011177HENRY     MANCINI        H345 POTOMAC AVE     JOHNSTON   IL....1997/03/24..
H011188RACHELLE  SULLIVAN        452 HOWARD STREET   SLIEMA     MA. ..1997/03/24..
H011190Justin    Case           S404 Lower Case BLVD Fresno     CA. ..1999/03/24..
H011199MABEL     SHIELDS        S232 GREEN STREET    WOODSTOCK  IL. ..1997/03/24..
H022211RALPH     JOHNSTON       H432 ADAMS DRIVE     MARENGO    IL....1997/03/24..
H022222GARY      GABLE          G567 HAYWARD AVE     FORT DODGEIA....1997/03/24..
```

Figure 19: Encoding/Decoding Data, Sample Segments

Copy the segments to sequential output. Encode the data using both the **KEY** value and the **PIN** code = `35`.

Control cards:

```
COPY INPCB(HRDB) OUTFILE(SYSUT2) PIN(35,KEY),
SCRAMBLE(1,0)
```

The **SCRAMBLE(1,0)** will encode the entire segment. By using the **KEY** option along with the **PIN** code, each segment will be encoded uniquely.

Output of the **COPY** provides statistics on the input and output data sets, number of segments copied and number of segments scrambled.

```
COPY COMMAND :    INPUT(HRDB)     OUTPUT(SYSUT2)
  OPENED HRDB     SEQ
  OPENED SYSUT2   LOAD   SYS02164.T093642.RA000.MX10002S.TEMP2.H01   LRECL=80,80      RECFM=F   DSORG=PS
  SEGMENTS READ: INPCB(HRDB)=44    COPIED: OUTFILE(SYSUT2)=44
  Data changed by the following operands:
   SCRAMBLE #4=44
COPY COMMAND COMPLETE; RETURN=0
```

Figure 20: **COPY** Command Output

Records now appear as:

```
O.......$.........$.................<..□|.!.Q.......!.kb.g........1.....C..C....
O...y...KK......03....0.........@'..3L..\..3.......jkl.........................
O...yy;...11111../111111111111....1.--..1_.\1111111_/.□/.1111%...3.............
O...zzx.x.......tx.m...........@.g.t...mn...n........xp.......t#...g=.g.fi.gh##
O.......v.u......vu.o..........o=@i.v.w.op~..p........ur.~.....v'..._.hh..h=...''
O.........-).\\\..&.-\\\\\\\\\\\....\.);;)..(<\...)).-.....*\\\&|..W.....E..E....
O.............................42.............................X..M..34i..i..XX
O...........EEEE... ...EEEEEEEE....E...... E...EEEEE........EE.....4....s..s....
```

Figure 21: Sample **SCRAMBLE** Segments

To **UNSCRAMBLE** (decode) these segments, the following control cards are coded:

```
COPY INFILE(SYSUT2) OUTPCB(HRDB) PIN(35,KEY),
UNSCRAMBLE(1,0)
```

After the **UNSCRAMBLE** using the control cards as shown above, the segments have been restored to their original value.

```
H011111BETTY    TOE            J555 PLAIN DR.       Denver    CO....1997/03/24..
H011121Bettyjo  BialoskiII     H123 Vaughn Way      XYZ
H011122JOHN     TOE            H555 PLAIN DR.       DENVER    CO....1985/11/14..
H011133NANCY    JONES          F751 OAK STREET      DENVER    CO...%1961/03/12..
H011144JOHN     JONES          S751 OAK STREET      DENVER    CO....2002/07/22..
H011155MATHEW   SMITH          J607 JEFFERSON STREETHARVARD   IL....1997/07/17..
H011166SALLY    JOHNSON        K352 SUPERIOR AVE    SHAKER HT OH. ..1965/01/31..
H011177HENRY    MANCINI        H345 POTOMAC AVE     JOHNSTON  IL....1997/03/24..
H011188RACHELLE SULLIVAN        452 HOWARD STREET   SLIEMA    MA. ..1997/03/24..
H011190Justin   Case           S404 Lower Case BLVD Fresno    CA. ..1999/03/24..
H011199MABEL    SHIELDS        S232 GREEN STREET    WOODSTOCK IL. ..1997/03/24..
H022211RALPH    JOHNSTON       H432 ADAMS DRIVE     MARENGO   IL....1997/03/24..
H022222GARY     GABLE          G567 HAYWARD AVE     FORT DODGEIA....1997/03/24..
```

Figure 22: Sample **UNSCRAMBLE** Segments

APPENDIX A: RETURN CODES

Return Code	Description	Further Information
0	Successful Completion.	See accompanying warning message.
4	Successful Completion with warning. If **COMPARE**, not all records compared successful.	See accompanying error message.
8	Unsuccessful completion (**Verify** mode). File Processing has not yet begun.	See accompanying error message.
12	Unsuccessful completion (**Processing** mode). The file processing may not be complete and the affected data files may need to be recovered.	See accompanying error message.
16	IMS Processing Error.	See accompanying error message.
20	Internal program failure. This error should be reported to MAX Software. The file processing may not be complete and the affected data files may need to be recovered. Missing SYSIN dd statement.	See accompanying error messages.
24	Problem with allocation of SYSPRINT file name.	

Note: These are the standard return codes, however they can be affected by the use of the installation defaults table to set return code values. In addition, the OPTION statement operands of **CBNFRC**, **COMPRC**, **EMPTYRC**, **MATCHKEYRC**, **NOSELRC**, **SETMAXRC**, **SETRC**, **TRANSRC**, and **TRUNCRC** can alter the return code values. See "Appendix D:Changing Installation Defaults" on page 119 for information on the installation defaults table.

APPENDIX B: COMMAND CROSS-REFERENCE

Operands	ACCUM	COMPARE	COPY	DELETE	DUMP	FPRINT	LIST	OPTION	PRINT	SKIP	SORT	UPDATE	CALCDAYS	Operand Description
ABEND	*	*	*	*	*	*	*		*	*	*	*	*	Abend when processing error.
ACCUM	*	*	*	*	*	*	*		*	*	*	*		Accumulate and print the total value for a field.
CALCAMT	*		*	*	*	*	*		*	*	*	*		Calculate new amount.
CALCDATE	*	*	*	*	*	*	*		*	*	*	*		Add/Subtract a day to a date type field.
CBNFRC								*						Set return code value for **FPRINT** copybook not found.
CHANGE	*		*	*	*	*	*		*	*	*	*		Change a record or segment in data file mode.
CODEPAGEIN			*		*		*			*				Set input code page for data transformation.
CODEPAGEOUT			*		*		*			*				Set output code page for data transformation.
COMPARE		*												Compare column ranges.
COMPFILE	*													Compare to file name.
COMPLIM		*												Alteration of match limit.
COMPPCB		*												Compare input from PCB.
COMPRC								*						Set return code value for **COMPARE** errors.
COMPRID		*												Initial record positioning for **COMPILE**.
COPYBOOK		*	*	*	*	*	*		*	*		*		COBOL layout data set name.
DELETE				*								*		Delete selected segment.
DUMP	*		*	*	*	*	*		*	*	*	*		Dump segment with record info and record key.
EDIT	*		*	*	*	*	*		*	*	*	*		Change a segment in text file mode.

Operands	ACCUM	COMPARE	COPY	DELETE	DUMP	FPRINT	LIST	OPTION	PRINT	SKIP	SORT	UPDATE	CALCDAYS	Operand Description
EMPTYRC								*						Set return code value for empty **INFILE**.
FORM	*	*	*	*	*	*	*		*	*	*	*		Control process, output format, and print options.
FORMAT			*				*							Data transformation output format.
FPRINT	*		*		*	*	*		*	*	*	*		Print record formatted using record layout.
FREQOUT	*		*	*	*	*	*		*	*	*	*		Specify output selection frequency.
GDATES													*	Specify Gregorian dates.
IF, AND, OR	*	*	*		*	*	*		*	*	*	*		Select specific records.
IN	*	*	*	*	*	*	*		*	*	*	*		Limit number records read from input file.
INEXIT	*	*	*	*	*	*	*		*	*	*	*		Provide record processing exit for **INFILE**.
INFILE	*	*	*	*	*	*	*		*	*	*	*		Read input from file name.
INPCB	*	*	*	*	*	*	*		*	*	*	*		Read primary input from PCB.
INRID		*												Initial record positioning for **INFILE**.
JDATES													*	Specify Julian dates.
LIMIT	*		*	*	*	*	*		*	*	*	*		Limit number of records written to primary output.
LIST	*		*	*	*	*	*		*	*	*	*		Print record.
MAPDD		*	*		*	*	*		*	*	*	*		File name of mapping criteria data set.
MATCH	*		*		*	*	*		*	*	*	*		Select records based upon a second file.
MATCHKEY	*		*	*	*	*	*		*	*	*	*		Select records by key on **INFILE** from value in **MTCHFILE**.
MATCHKEYRC								*						Set return code value if mismatch on **MATCHKEY**.
MOVE			*	*	*	*	*		*	*	*	*		Build a record.
MTCHFILE	*		*	*	*	*	*		*	*	*	*		Match file name.

Operands	ACCUM	COMPARE	COPY	DELETE	DUMP	FPRINT	LIST	OPTION	PRINT	SKIP	SORT	UPDATE	CALCDAYS	Operand Description
NOSELRC								*						Set return code value for no records selected.
OUT		*	*	*	*	*	*		*	*	*	*		Maximum number of records to output.
OUTFILE			*		*	*	*		*	*	*			Write output to file name.
OUTPATH			*											Specify UNIX file name.
OUTPCB	*		*	*	*	*	*		*	*	*	*		Write output to PCB.
PADCHAR	*		*	*	*	*	*		*	*	*	*		Initialize any part of record not initialized.
PDSSTAT			*									*		Maintain ISPF directory statistics.
PIN	*		*		*	*	*		*	*	*	*		Build encryption.
PLICOPY		*	*	*	*	*	*		*	*	*	*		PL/I layout data set name.
PRINT	*		*	*	*	*	*		*	*	*	*		Print record with record information and record key.
REFDD			*	*	*	*	*		*	*	*	*		File name of reformat criteria.
REPLACE	*		*	*	*	*	*		*	*	*	*		Overlays data in a record.
RID	*		*	*	*	*	*		*		*	*		Initial record key positioning for all file types.
SCRAMBLE	*		*	*	*	*	*		*	*	*	*		Encode data.
SELECT	*		*	*	*	*	*		*	*	*	*		Select every n^{th} record.
SETMAXRC								*						Set maximum return code to value.
SETRC	*	*	*	*	*	*	*	*	*	*	*	*		Set specific return code.
SKIP	*	*	*		*	*	*		*			*		Bypass selected number of records. **Note:** This operand can be used with the '**ALL**' or '**MEM**' version of the command only.
SORTPARM											*			Pass parameters to **SORT**.
STOP	*	*	*	*	*	*	*		*	*	*	*		Stop processing at selected record.

Operands	ACCUM	COMPARE	COPY	DELETE	DUMP	FPRINT	LIST	OPTION	PRINT	SKIP	SORT	UPDATE	CALCDAYS	Operand Description
STOPMAXRC								*						Set condition to bypass remaining commands when max return code meets condition.
STOPRC								*						Set condition to bypass remaining commands when return code meets condition.
SUBSYS	*	*	*	*	*	*	*		*	*	*	*		Subsystem name for copybook retrieval.
TRANSLATE	*		*	*	*	*	*		*	*	*	*		Translate characters.
TRANSRC								*						Set return code value for transformation errors.
TRUNCRC								*						Sets return code value for record truncation.
UNSCRAMBLE	*		*	*	*	*	*		*	*	*	*		Decode scrambled data.
WARNMATCHKEY								*						Set warning of MATCHKEY errors OFF or set warning limit.
WARNTRANERR								*						Set warning of transformation errors OFF or set warning limit.
WRITE	*		*	*	*	*	*		*	*	*	*		Write specific selected record to file name.

APPENDIX C: COPYBOOK SUPPORT

COBOL Copybooks are supported as follows:

- Field names to 40 bytes in length.
- REDEFINES up to 10 levels.
- OCCURS up to 32760 occurrences.
- USAGE types of:

BINARY	
COMPUTATIONAL	**COMP**
COMPUTATIONAL-1	**COMP-1**
COMPUTATIONAL-2	**COMP-2**
COMPUTATIONAL-3	**COMP-3**
COMPUTATIONAL-4	**COMP-4**
DISPLAY	
PACKED-DECIMAL	

PIC clause characters supported:

- **B 0 / , . + - CR DB Z * $ 9 A X S V P**
- Duplicate field names supported as FILLER.

PL/I Copybooks are supported as follows:

- Field names up to 31 characters.
- Multi dimensional arrays with up to 9999 occurrences.

Data types as follows:

- **BIN** **BIN FLOAT** **BINARY FIXED** **BITS**
  CHAR **DEC** **DEC FIXED** **DEC FLOAT**
  FIXED **FIXED BINARY** **FIXED DEC** **FLOAT**
  FLOAT BIN **FLOAT DEC** **PICTURE**

Batch Copybook Print Feature

A feature is available to print expanded copybook information. This feature in invoked by the **PRINT** command. This command, in combination with the following operands, will provide a report with information about the copybook. This report includes group field indicators and aggregate lengths, **OCCURS DEPENDING ON** fields expanded to the maximum number of occurrences, field formats, field position, and **REDEFINES** information.

Operands Supported:

COPYBOOK|PLICOPY	use to provide data set and member name. This operand is REQUIRED.
OUTFILE	use to direct report to a specific data set. This operand is optional. If this operand is not used, the report will be directed to the SYSLIST output.
SUBSYS	Use this operand if a subsystem is used to process the copybook.

Examples:

Print copybook information to SYSLIST.

```
PRINT COPYBOOK(MXS.TEST.COPYLIB(MEMNAME))
```

Direct the output of the print to a data set.

```
PRINT COPYBOOK(MXS.TEST.COPYLIB(MEMNAME)),
OUTFILE(SYSUT2)
```

APPENDIX D: CHANGING INSTALLATION DEFAULTS

The member MAXDFLTS, in the installation JCL library, is available for the user to change defaults as they exist in the MAX/Batch product. As shipped, this member contains the standard defaults for each of the options that can be changed.

To run the product with the standard defaults, no action is required. To change one or more of the standard defaults, edit the member, make the changes as noted, and submit the job. Once this module is linked into the load library containing the MAXIBAT module, the new options will be in effect..

Option	Standard default
Input data file name	SYSUT1
RC - records truncated during **COPY**	4
RC - no records selected from input	4
RC - input file is empty	8
RC - not all records compare	4
RDW Value	3
RC - transformation errors	4
RC - missing keys from **MATCHKEY**	4
Eliminate warning for transformation errors or set warning level	20
Eliminate **MATCHKEY** warnings or set warning limit	20
Character translation	DEFAULT (USA English)[1]
Lines per page	60
Illogic indicator[2]	N
Input disposition	C = close between commands
Output disposition	C = close between commands
RC - if no copybook found with **FPRINT** command[3]	
Number of external Translate tables[4]	3
Table name and pointer[4]	3 default tables
Translate tables[4]	256 byte tables

1.Valid character sets include CP870 (Czech) and CP838 (Thai).

2.Use this indication to receive error (Y) if illogical operand sequences are found. If an OR follows and operand other than another OR, an IF or an AND it is considered illogical. The OR in this series will begin a new IF sequence. Set the indicator to Y to detect these situations.

3.This applies only to the **FPRINT** command. If a copybook is not found and the segment is printed in DUMP format, a special return code can be set.

4.The placement and coding of the translate tables must follow the example shown in the MAXDFLTS member that is shipped with MAXIBAT. The fullword value in NUM#TRT must indicate the number of tables (full 256 byte tables) that follow. This fullword is followed immediately with each 8 byte table name and its 4 byte address pointer. Follow this with the full 256 byte translate tables. The tables shipped with this are examples that can be used as is or modified by the user.

Temporary Overrides of Default Values

It is possible to override both return codes and warning values on a temporary basis. This can be done by using the **OPTION** statement to set the return code values and warning conditions. Refer to the following operands for details on setting the overrides: **CBNFRC**, **COMPRC**, **EMPTYRC**, **MATCHKEYRC**, **NOSELRC**, **SETMAXRC**, **SETRC**, **TRANSRC**, and **TRUNCRC**.

Return codes are determined in the following order:

1. Temporary return codes always take precedence.

If not set:

2. Installation defaults from MAXDFLTS table.

If not found:

3. MAXIBAT default return codes. See standard default above.

APPENDIX E: SECURITY CONSIDERATIONS

The MAX IMS/UTIL product honors all standard security interfaces. For example, if a user attempts to use the product to update a data set to which they have been denied access, the following screen will appear:

```
ICH408I USER(your_userid ) GROUP(xyz     ) NAME(####################)
  SYS1.SAMPLIB CL(DATASET ) VOL(Z4RES1)
  INSUFFICIENT ACCESS AUTHORITY
  FROM SYS1.* (G)
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(READ   )
IEC150I
913-38,IFG0194E,your_userid,ISPFPROC,ISP14273,0A80,Z4RES1,SYS1.SAMPLIB
***
```

Figure 23: Security considerations panel

If the user attempts to update a database with a command for which the PSB PROCOPT does not allow, IMS will reject the operation and a message will be displayed.

Standard IMS AGN security will enforce access to a database with only PSBs to which the user is authorized when running in BMP mode.

INDEX

D

E

F

G

I

J

L

M

N

O

P

R

S

T

U

W

X